MATLAB, the MATLAB Web Server, and the Database Toolbox

StanAhalt
Academic Lead

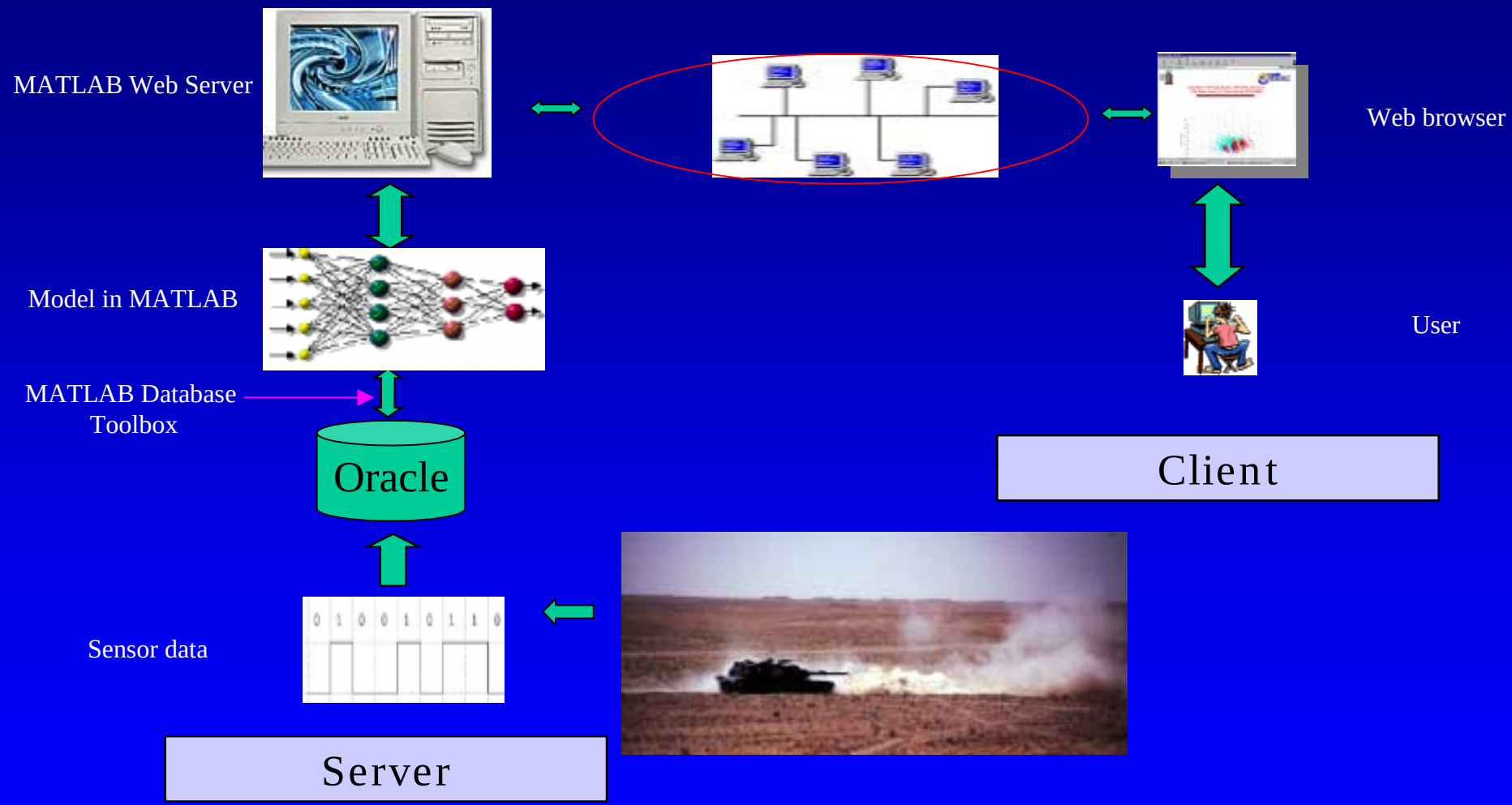
Ashok Krishnamurthy
Academic Associate
Ohio State University

sca@ee.eng.ohio-state.edu
krishnamurthy.1@osu.edu

Motivations

- MATLAB is omnipresent in the engineering community, as is the web.
  - How can we put our MATLAB applications on-line easily?
    - **MATLAB Web Server**
- What about data?
  - Larger and larger data sets are being used.
  - How can we use MATLAB to access these data sets?
    - **MATLAB Database Toolbox**.

Application prototype



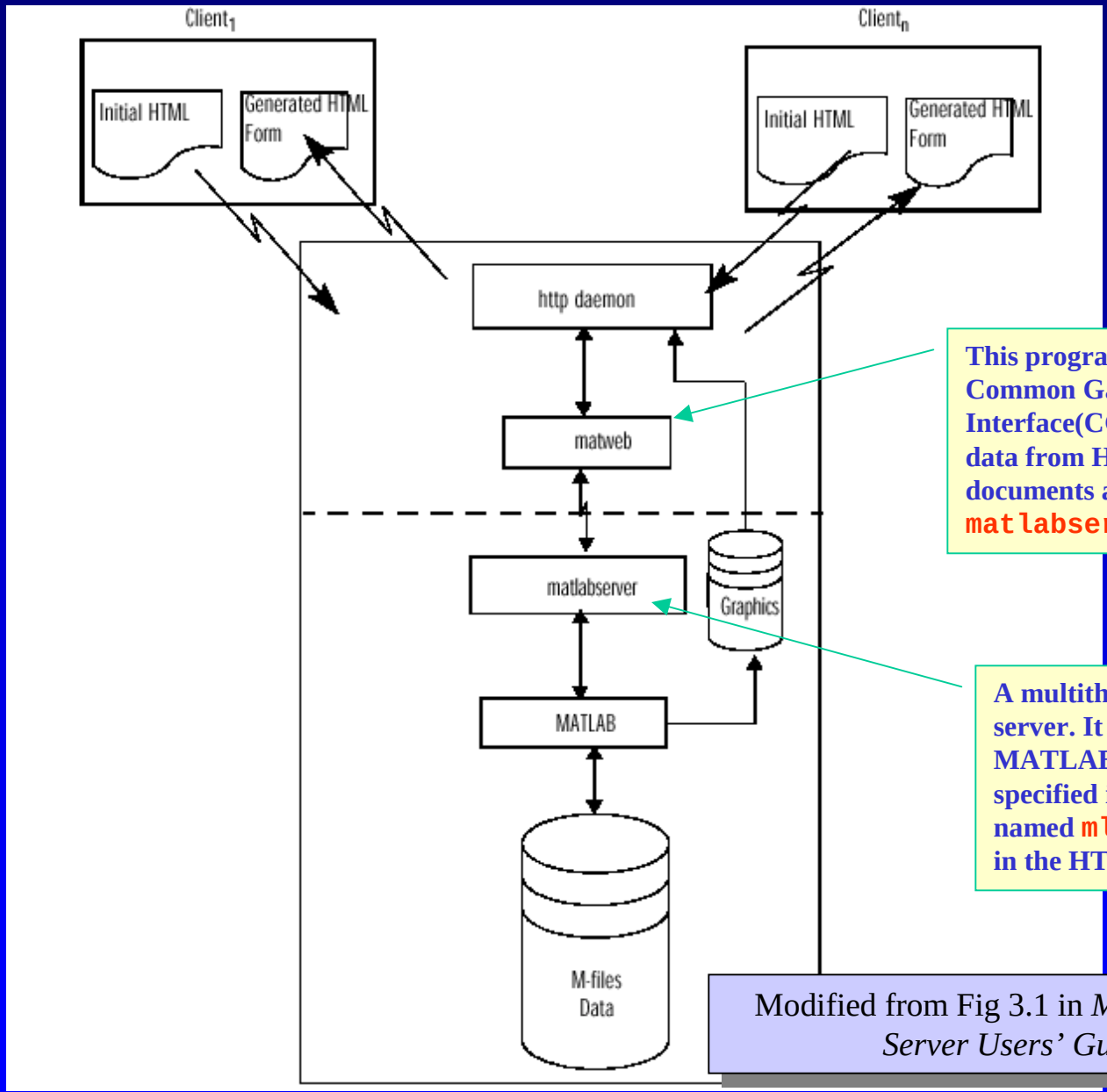
MATLAB Web Server and Database Toolbox make the above application not only possible, but easy.

MATLAB Web Server

MATLAB on the web

- What is the MATLAB Web Server?
  - Interface between the web and MATLAB
  - Makes use of the web as a GUI for the user to specify/parameterize inputs, and view output(s), including graphics.
  - MATLAB runs on the server which accepts the user requests from the client and transfers the results to the web on the client side.
- Platforms supported
  - Windows NT(not sure about Windows 2000, trials underway)
  - UNIX(Solaris)
  - Linux

Overview

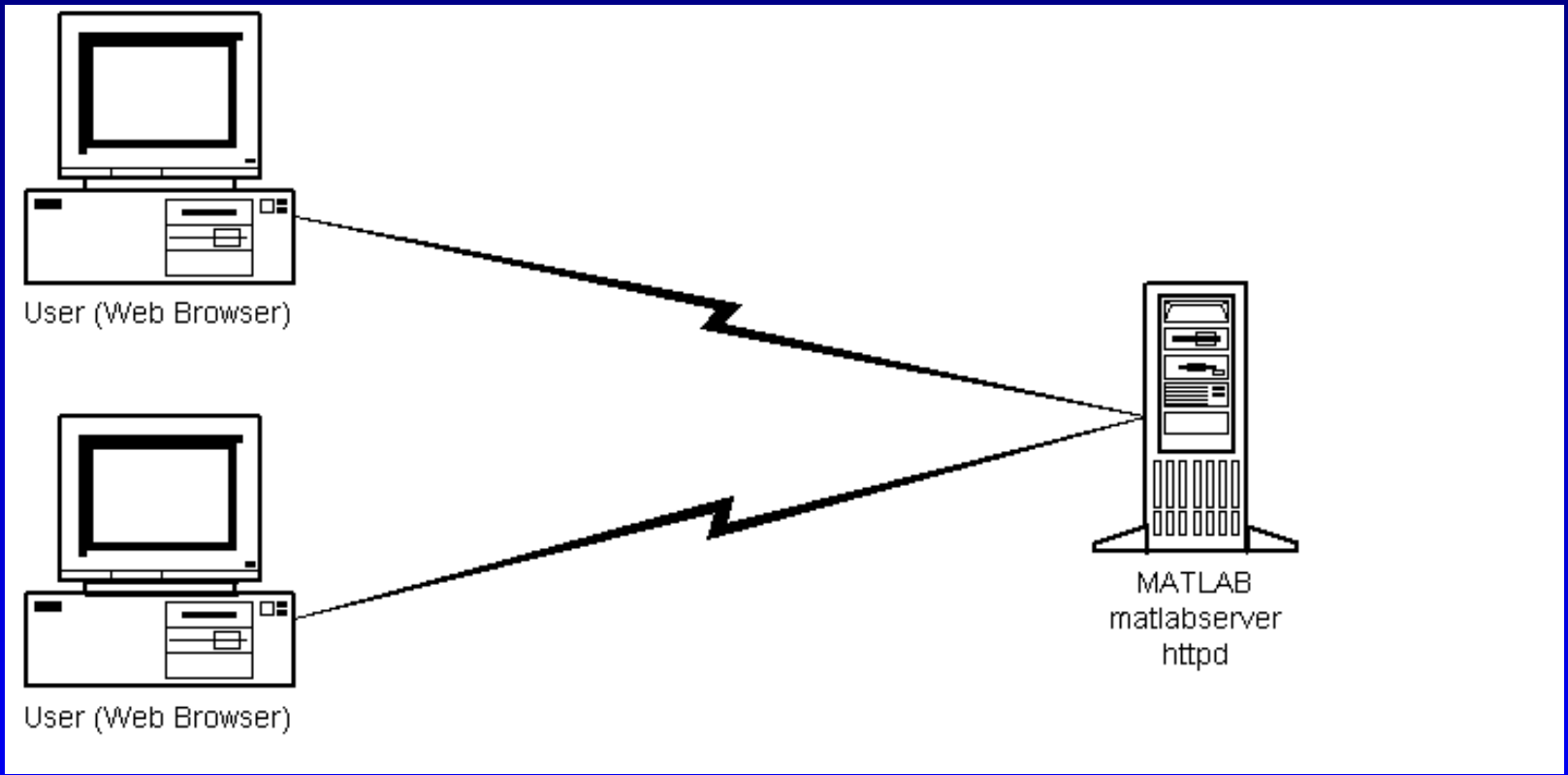


This program uses the Common Gateway Interface(CGI) to extract data from HTML documents and transfer it to `matlabserver`

A multithreaded TCP/IP server. It runs the MATLAB program (M-file) specified in a hidden field named `mlmfile` contained in the HTML document.

Modified from Fig 3.1 in *MATLAB Web Server Users' Guide*

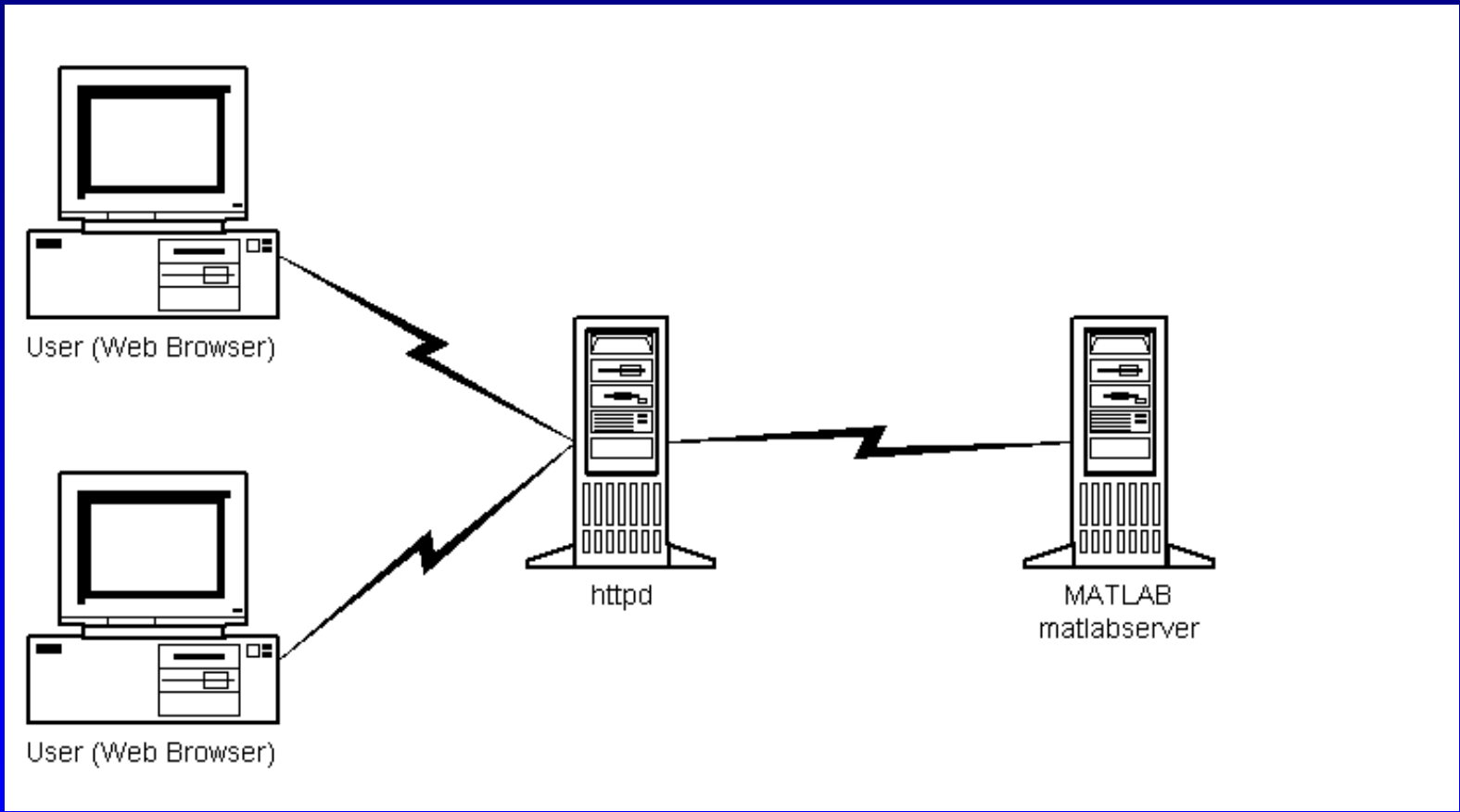
Configuration 1



In the simplest configuration, a Web browser runs on your client workstation, while MATLAB, the MATLAB Web Server (matlabserver), and the Web server daemon (httpd) run on another machine.

From page 1-2 in *MATLAB Web Server Users' Guide*

Configuration 2

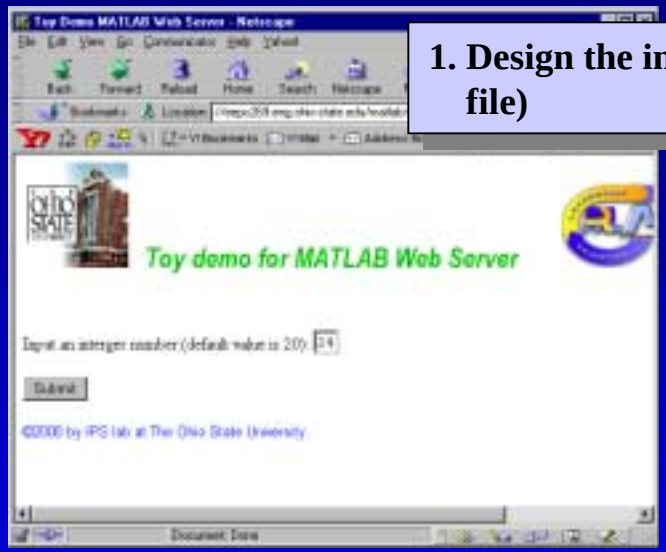


In a more complex network, the Web server daemon can run on a separate machine.

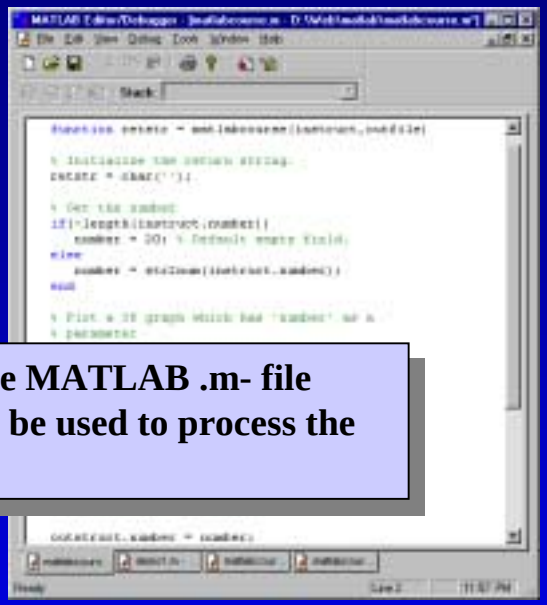
From page 1-3 in *MATLAB Web Server Users' Guide*

The 4 steps in the design of a MATLAB Web Server Application

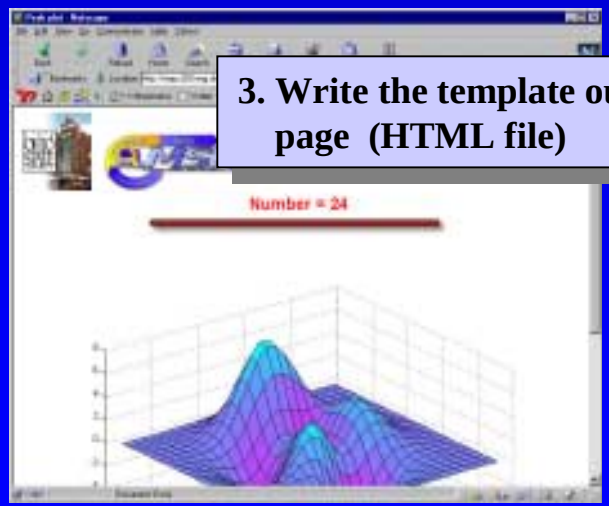
1. Design the input page (HTML file)



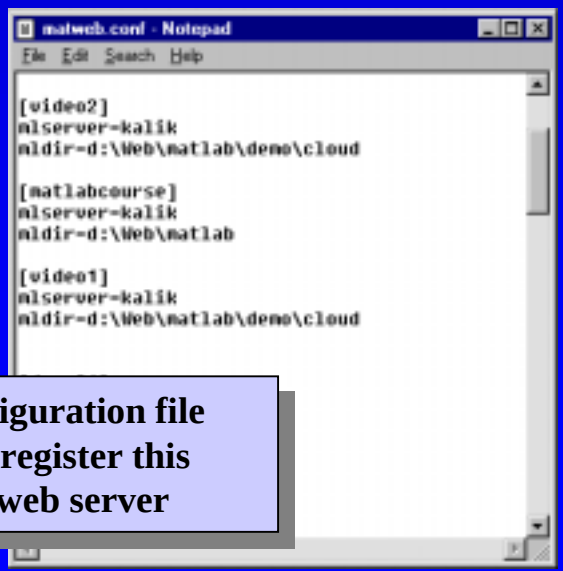
2. Code the MATLAB .m- file which will be used to process the data



3. Write the template output page (HTML file)



4. Modify the configuration file matweb.conf to register this application to the web server



Example(1) -- Input page

The input page is the form that users will use to interact with the MATLAB webserver to specify requests, provide data, set parameters, etc. It is written in html

```
<pre><img SRC="logo-rt.gif" height=100 width=100>  <b><i><fo
<font face="Arial Narrow"><font size=+3>Toy demo for MATLAB Web S
</font></font><font size=+4> </font></font></i></b><img SRC=

<form action="/cgi-bin/matweb.exe" method="POST">
    <input type="hidden" name="mlmfile" value="matlabcourse">
    <p>Input an interger number:(default value is 20): 
     <input type="text" size="2" maxlength="2" name="number">
    <p><input type="submit" name="Submit" value="Submit">
<br></form>

<p><font face="Arial"><font color="#0000FF"><font size=-1>&copy;20
IPS lab at The Ohio State University.</font></font></font>
</body>
```

Common HTML statements

call MATLAB web server!

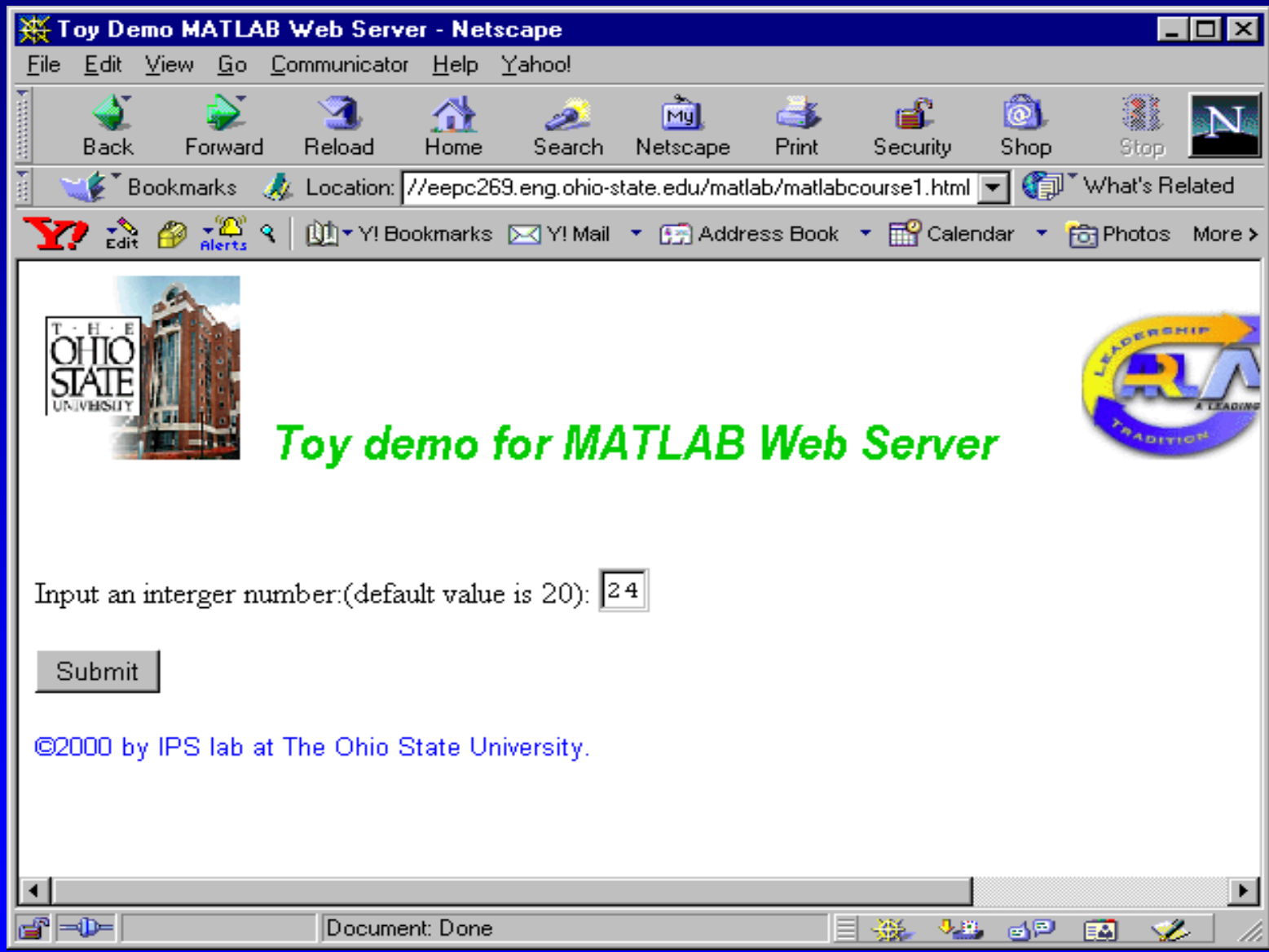
M-file name that will be executed!

Parameters passed to code!

Users can design this page using an editor, or Front Page or …

The NAME attribute is a required field and is used to identify the data for the field. The VALUE attribute specifies the value. For this hidden input, mlmfile is the name for this input and matlabcourse is the currently specified value for this input.Note that this input is not explicitly used in the m-file. The designer must keep the name of this input as mlmfile, and assign the name of the m-file as the value of this input.

Example – Rendered Input page



wscleanup deletes file ‘result.jpg’ if it exists and are older than 1 hour.

Example(2) -- M-file

```
function retstr = matlabcourse(instruct,outfile)

% Initialize the return string.
retstr = char('');

% Get the number
if(~length(instruct.number))
   number = 20; % Default empty field.
else
   number = str2num(instruct.number);
end

% Plot a 3D graph which has 'number' as a
% parameter

cd(instruct.mldir);
wscleanup('result.jpg',1);
f=figure('visible','off');
P = peaks(number);
C = del2(P);
surf(P,C)
colormap cool;
wsprintjpeg(f,'result.jpg');

% Assign values to output parameters

outstruct.number = number;
outstruct.GraphFileName = sprintf('result.jpg');
outstruct.date = date;

templatefile = which('matlabcourse2.html');
if (nargin == 1)
   retstr = htmlrep(outstruct, templatefile);
elseif (nargin == 2)
   retstr = htmlrep(outstruct, templatefile, outfile);
end
```

Parse the input parameters

Processing steps

Note use of wsprintjpeg

Assign output parameters (see webserver users guide)

Generate the output page

Substitute values for variable names in HTML document.

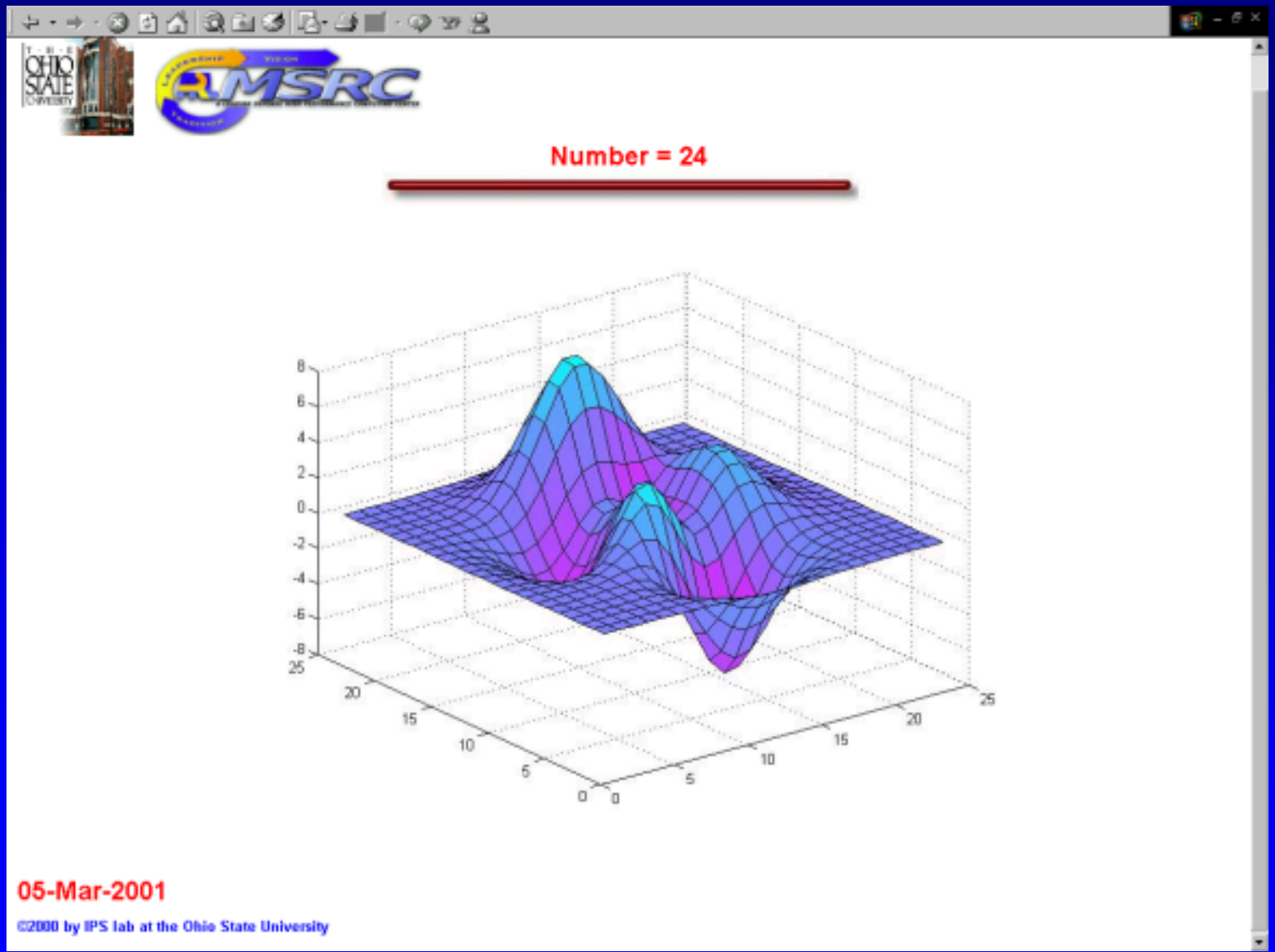
checking ‘nargin’: If the number of input parameters is 2 the m-file will write the final output page to ‘outfile’ so that you can open that file to check the results. This serves as an aid to debugging.

Example (3) – template output page

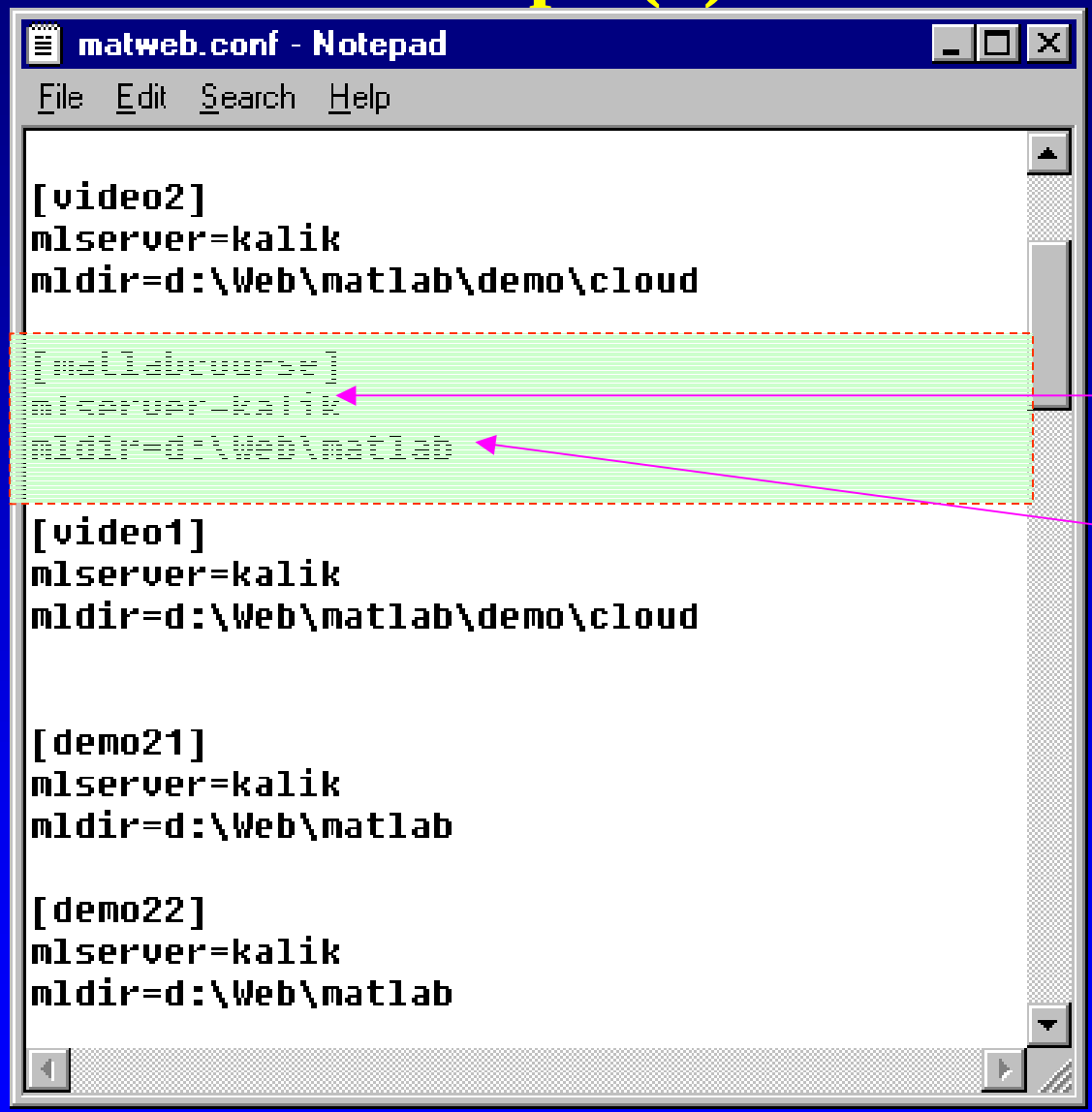
```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.74 [en] (WinNT; U) [Netscape]">
   <title>Peak plot</title>
</head>
<body bgcolor="#FFFFFF">
<img SRC="/matlab/logo-rt.gif" height=110 w                    ; 
<img SRC="/matlab/arl.gif" height=83 width=2

<center><b><font face="Arial"><font color="#FF0000">
<font size=+2>Number = $number$
<br><img SRC="/matlab/bar.jpg" height=30 width=450>
<p><img SRC="/matlab/$GraphFileName$" BORDER=0 ></center>
<p>$date$

<br><font face="Arial"><font color="#0000FF"><font size=-1>&copy;2000 by
IPS lab at the Ohio State University</font></font></font>
</body>
</html>
```

The variables delimited by $ will be replaced by the processed result

Rendered output page



Example(4) -- matweb.conf



Machine name where MATLAB Web Server resides

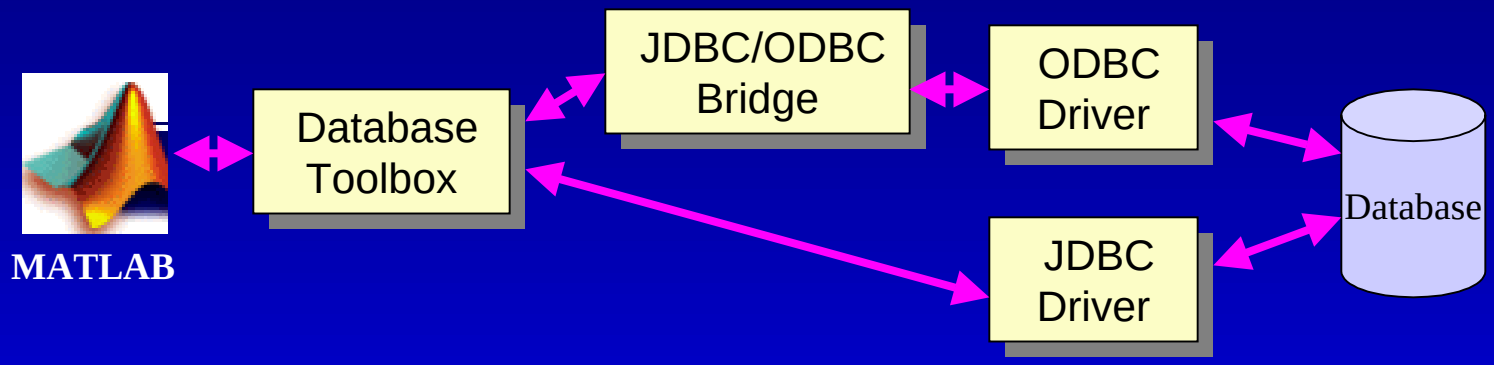
Working directory

/cgi-bin is the directory where the Common Gateway Interface (CGI) programs reside. Matweb.exe is one such program.

Note: file matweb.conf is located in the directory /cgi-bin or equivalnent.

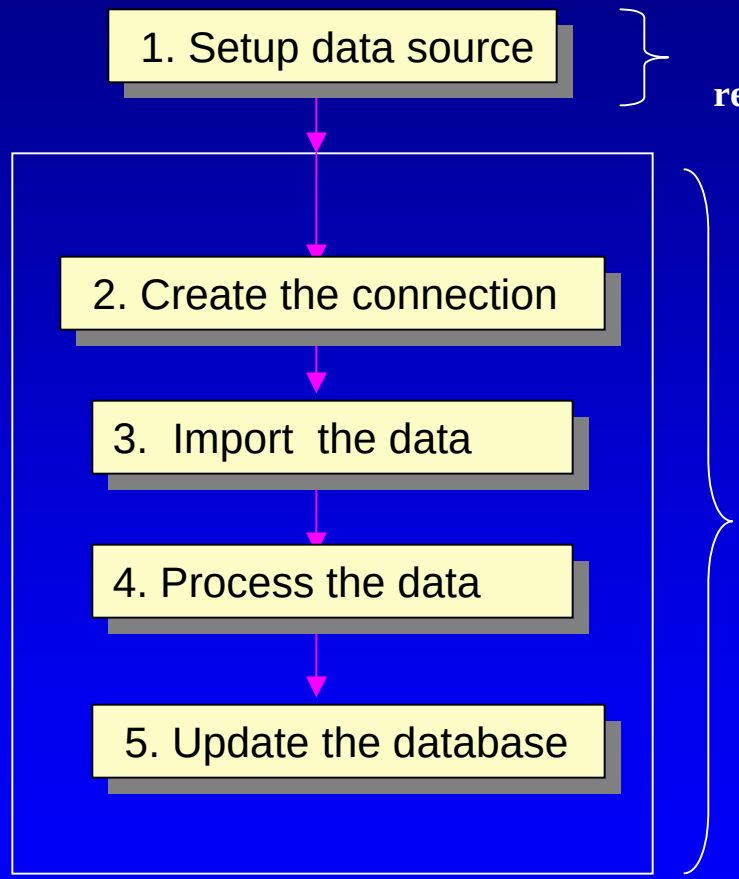
MATLAB Database Toolbox

Accessing Database from MATLAB



- MATLAB Database Toolbox Interface between MATLAB and database
  - For PC platforms, the Database Toolbox supports Open Database Connectivity(ODBC) drivers used with the supported databases.
  - For UNIX and PC platforms, the Database Toolbox supports Java Database Connectivity (JDBC) drivers.
  - The Database Toolbox supports SQL commands.

Using the MATLAB Database Toolbox

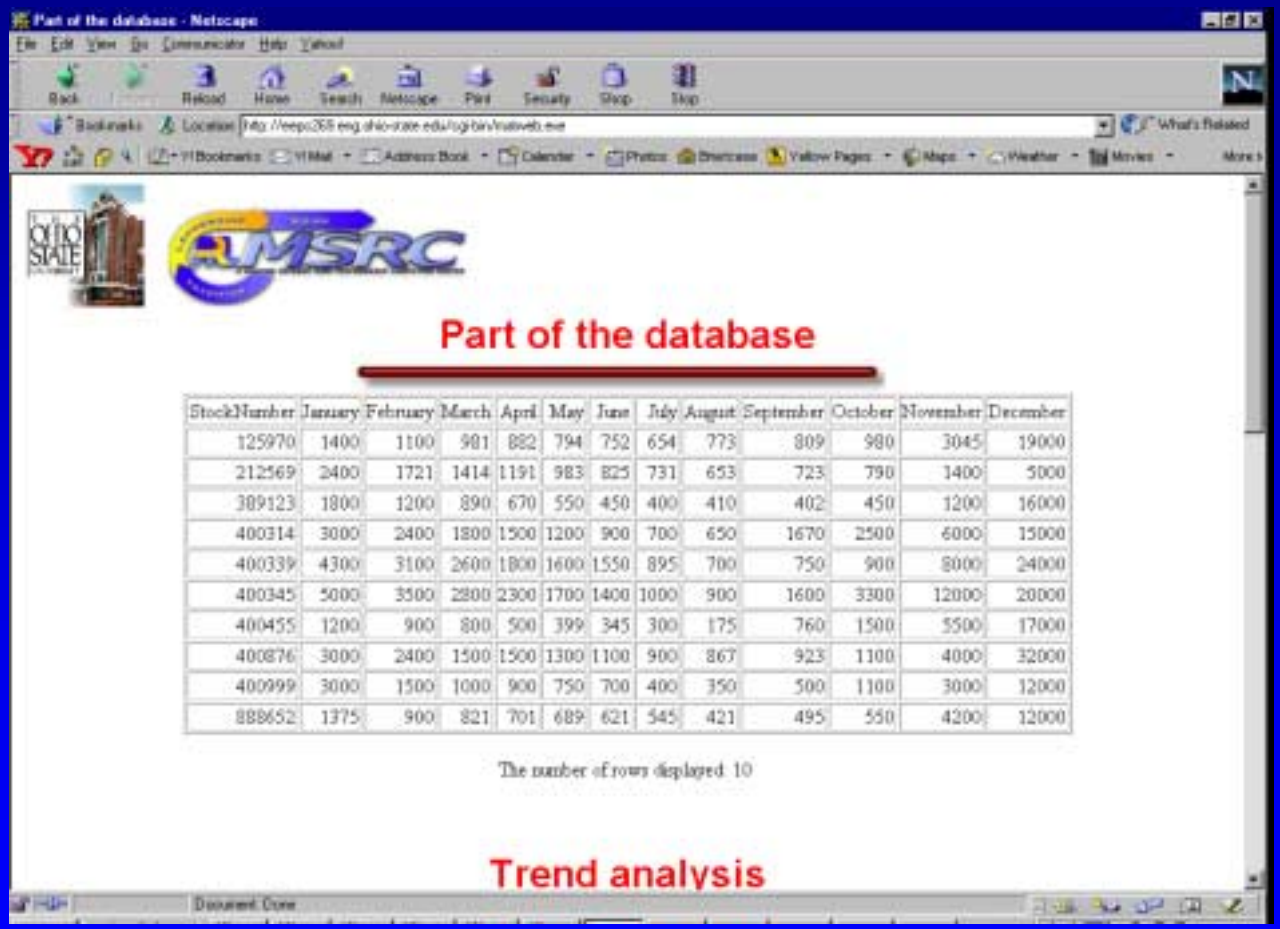


Do this outside MATLAB (Using operating system: remote|local, database_name, user access, etc.)

In your M-file

Example

Http://eepc269.eng.ohio-state.edu/matlab/demo1-1.html



StockNumber	January	February	March	April	May	June	July	August	September	October	November	December
125970	1400	1100	981	882	794	752	654	773	809	980	3045	19000
212569	2400	1721	1414	1191	983	825	731	653	723	790	1400	5000
389123	1800	1200	890	670	550	450	400	410	402	450	1200	16000
400314	3000	2400	1800	1500	1200	900	700	650	1670	2500	6000	15000
400339	4300	3100	2600	1800	1600	1550	895	700	750	900	8000	24000
400345	5000	3500	2800	2300	1700	1400	1000	900	1600	3300	12000	20000
400455	1200	900	800	500	399	345	300	175	760	1500	5500	17000
400876	3000	2400	1500	1500	1300	1100	900	867	923	1100	4000	32000
400999	3000	1500	1000	900	750	700	400	350	500	1100	3000	12000
888652	1375	900	821	701	689	621	545	421	495	550	4200	12000

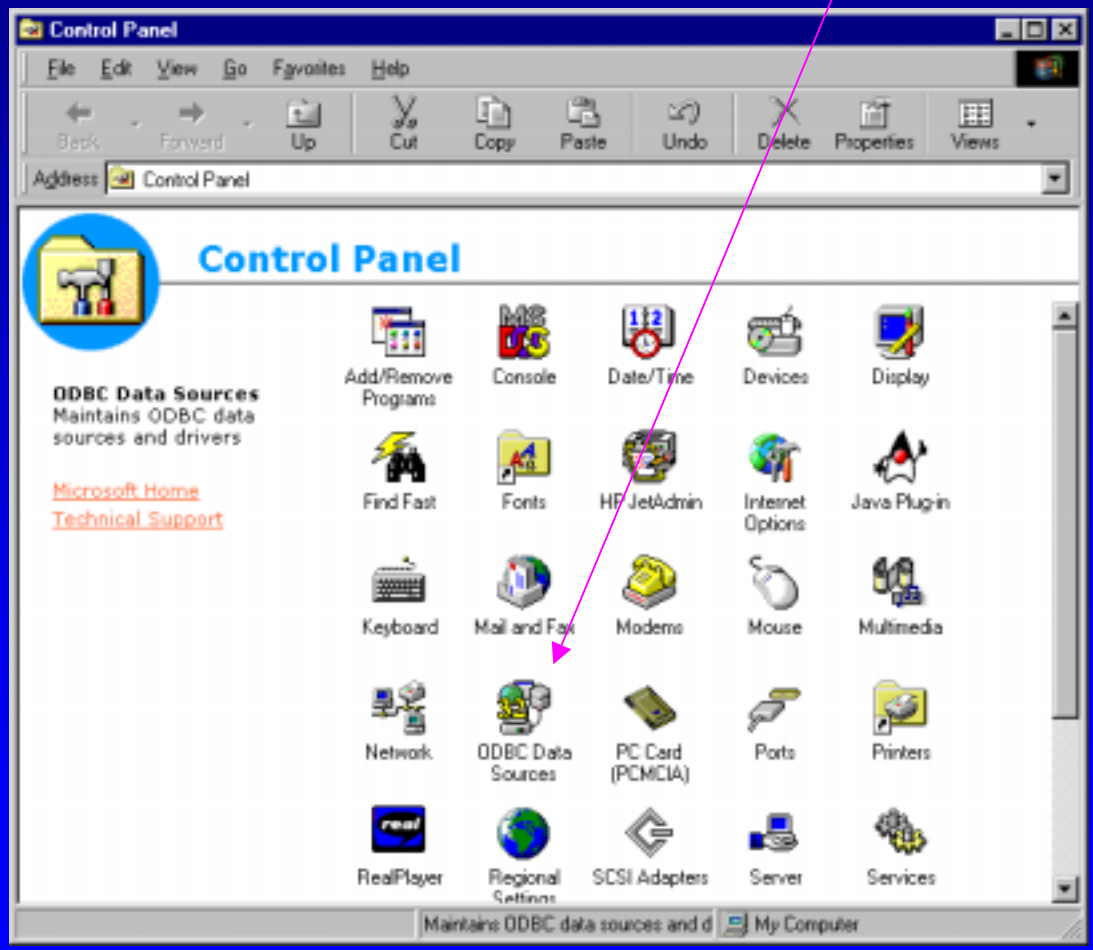
The number of rows displayed: 10

This example is implemented using the MATLAB Database Toolbox and the MATLAB Web Server.

- It reads part of one example database.
- It offers a graphical visualization of the data and then processes the data.
- The database can be subsequently updated with processed results, and the new content of the database can be redisplayed.

Step 1(a): Setup data source

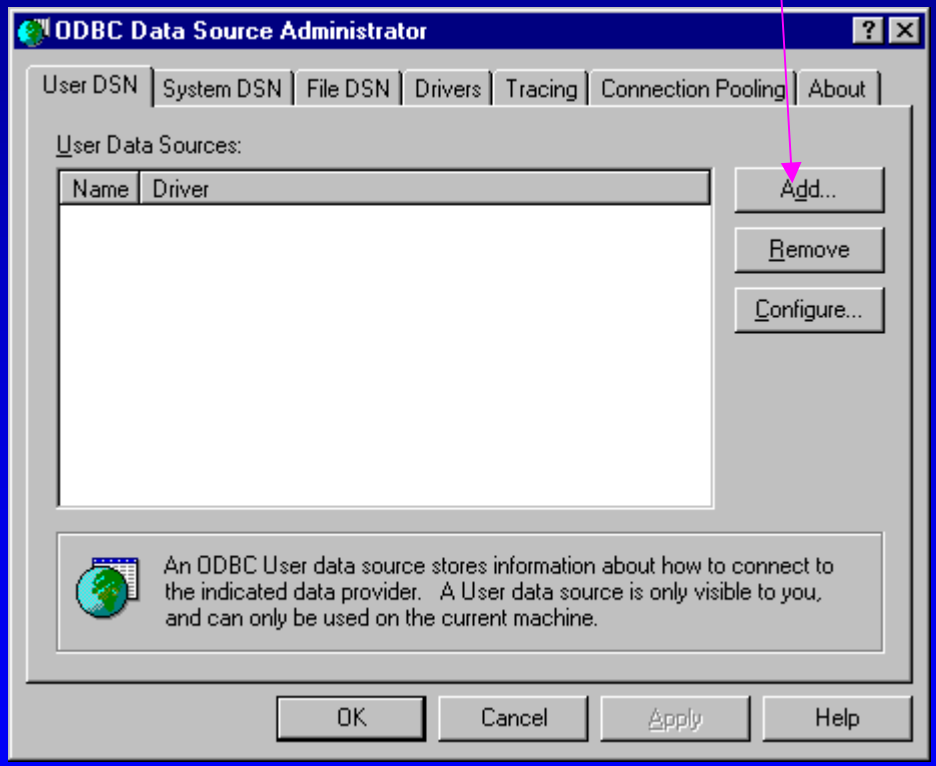
1: Invoke control panel and double click ODBC Data Sources.



This example was implemented on a WinNT platform.

Step 1(b): Setup data source

2: Select the User DSN tab and click Add.

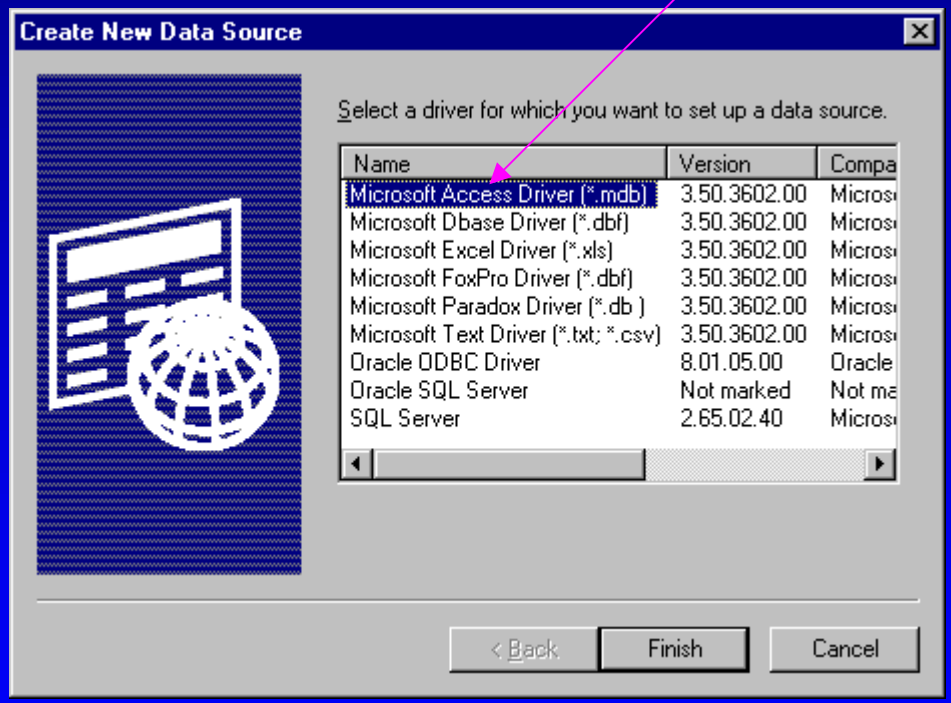


If the database is to be served over the web, then the database must be registered under a system DSN (see later example)

This example was implemented on a WinNT platform.

Step 1(c): Setup data source

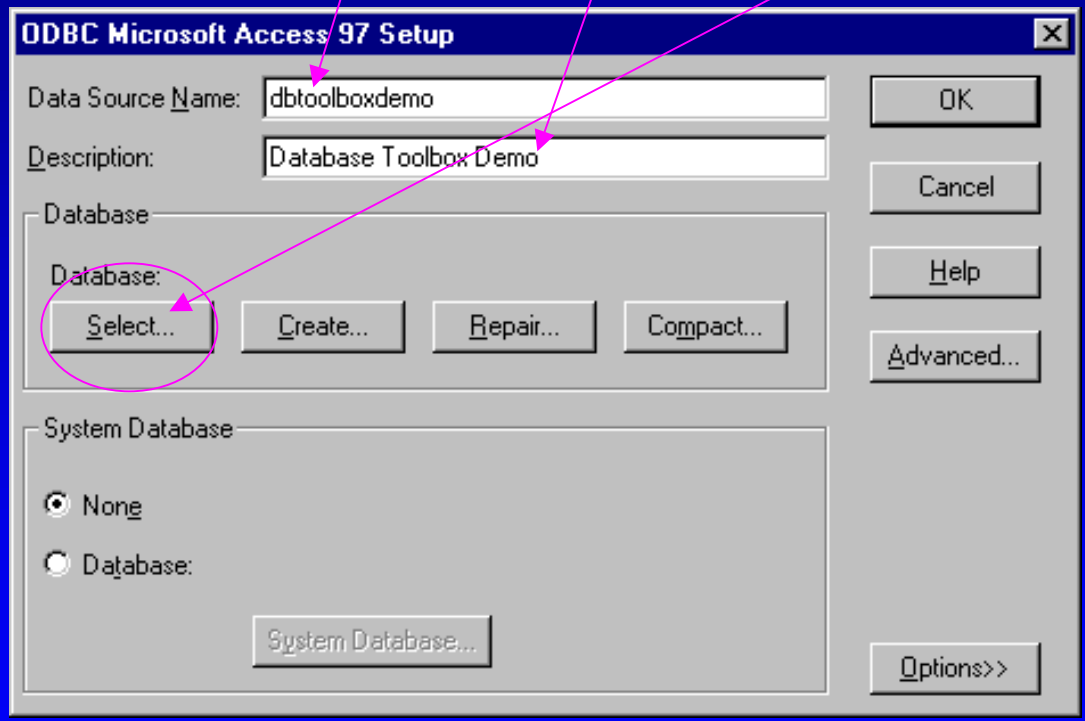
3: Select the ODBC driver. In this case, choose Microsoft Access Driver.



Compared with Oracle, Microsoft's Access Database is relatively easier to set up. Here, we use Microsoft Access database as an example.

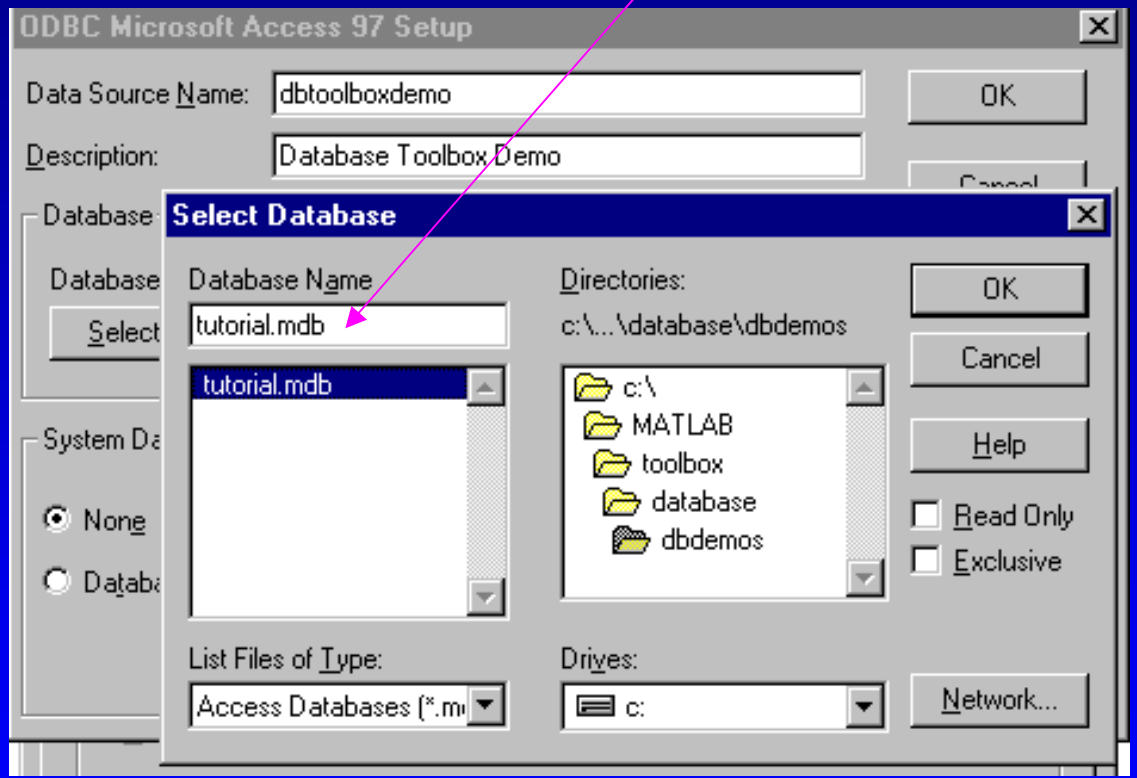
Step 1(d): Setup data source

4: Provide a Data Source Name and Description, then click select.

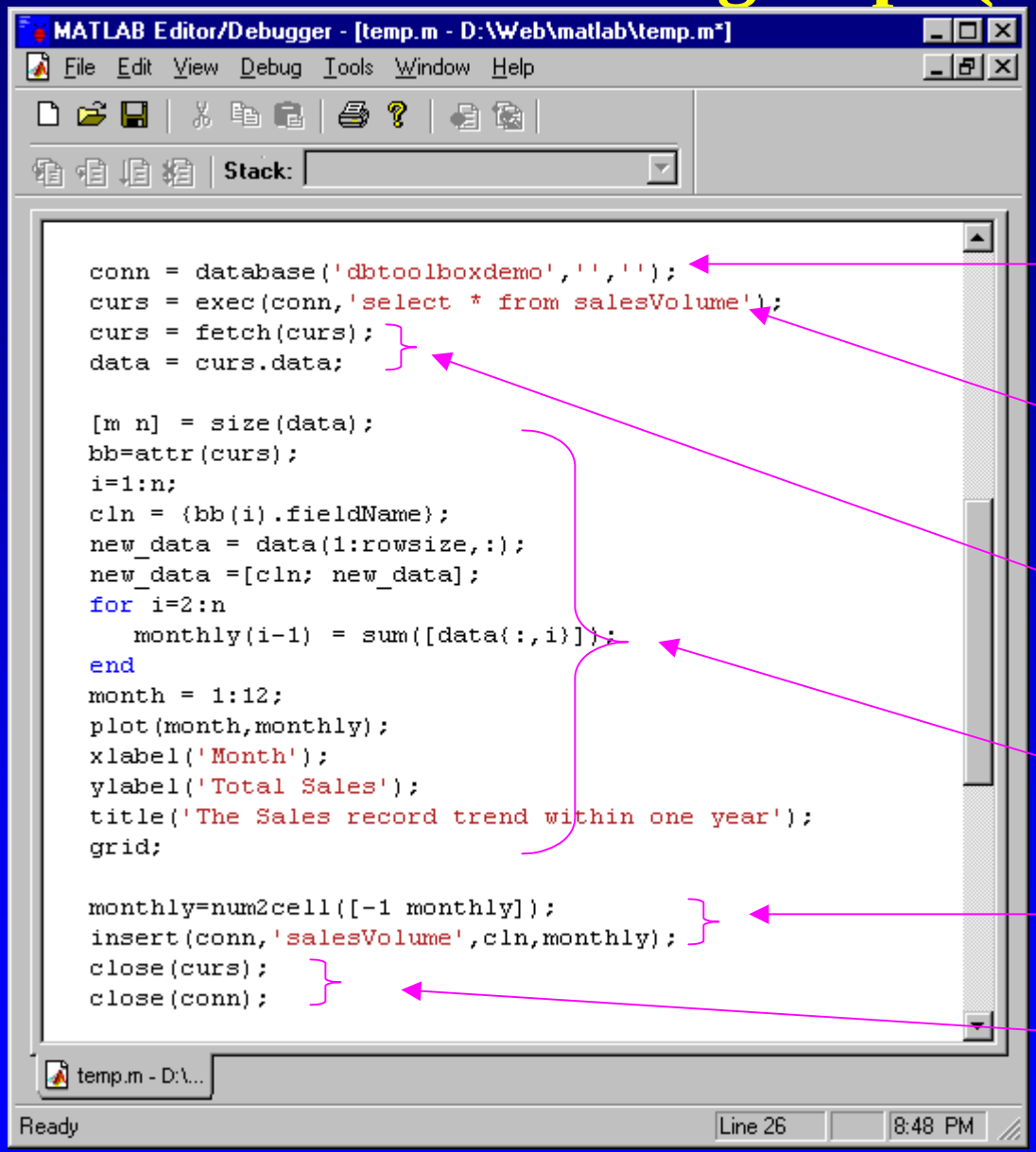


Step 1(e): Setup data source (associate file with database name)-- done!

5: Select the database file.



Processing steps (in M-file)



construct the connection with the target database

Execute the SQL query. The exec function returns a cursor (pointer) object.

Import the data. The new cursor object contains the rows of data retrieved.

Other processing to manipulate the data

Database update

Close the connection

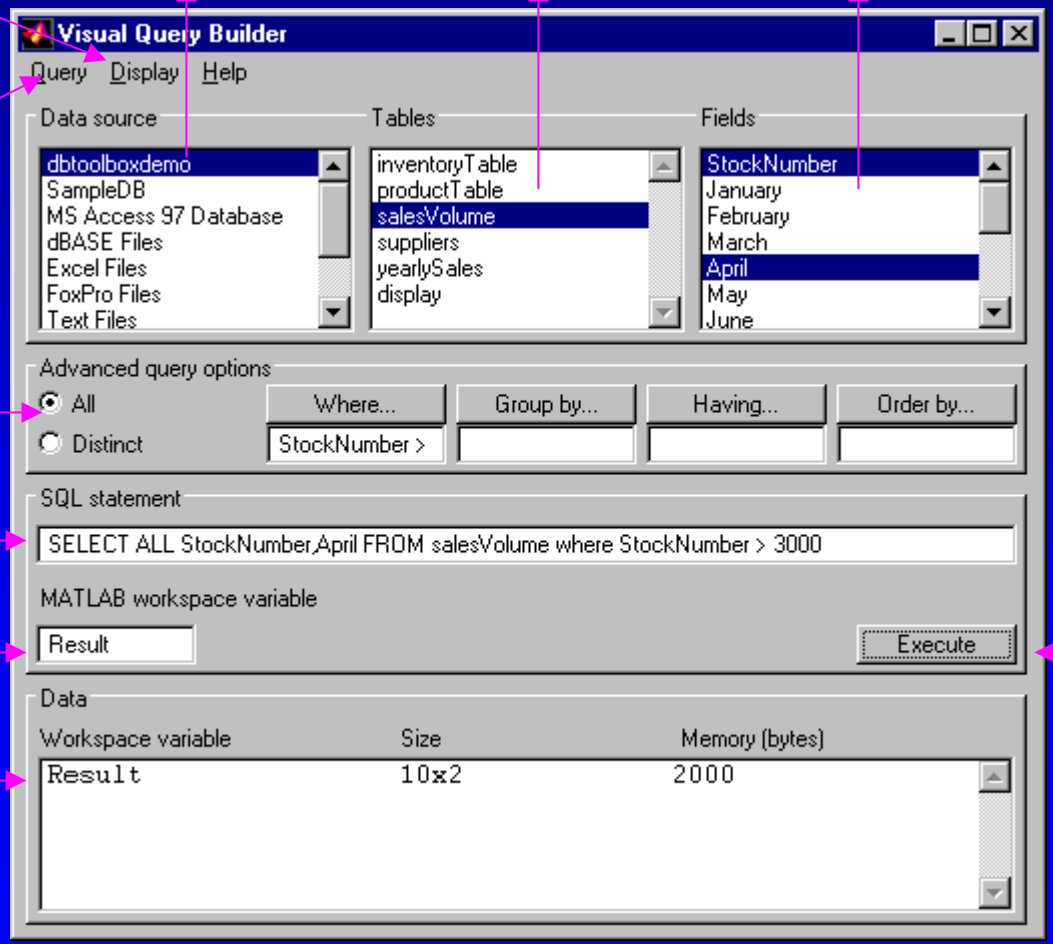
Visual Query Builder

- The Visual Query Builder (VQB) is an easy-to-use graphical user interface for retrieving data from your database.
- When should this be used?
  - If you want to retrieve information from relational databases for use in MATLAB and you are not familiar with the Structured Query Language (SQL) and database applications.
  - If you are familiar with SQL and your database applications, use the Visual Query Builder to build SQL queries easily and to import results into MATLAB, or use Database Toolbox functions instead.

Visual Query Builder (cont.)

```
>> querybuilder
>>
```

Type querybuilder to run Visual Query Builder



9: View query results in table, chart, and report formats.

1: Select the data source.

2: Select the tables.

3: Select the fields you want to retrieve.

10: Save, load, and run queries.

4: Refine the query, if needed.

5: View the SQL statement.

6: Assign a variable for the results.

8: Double-click to view query results in the MATLAB command window.

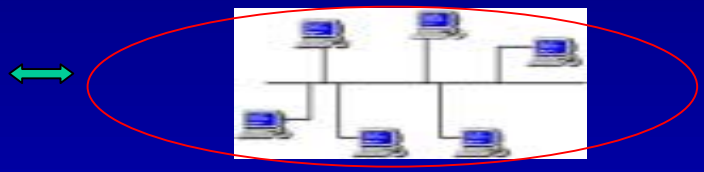
7: Run the query.

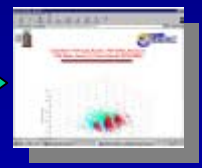
MATLAB, the Web, and a Database --- a foundation for Data Mining

A more complete example

FFT Operation and Visualization on Database Data

MATLAB Web Server





Web browser

Model in MATLAB

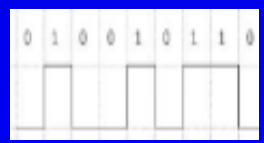
FFT

User

MATLAB Database Toolbox

Oracle

Client



Sensor data

Server

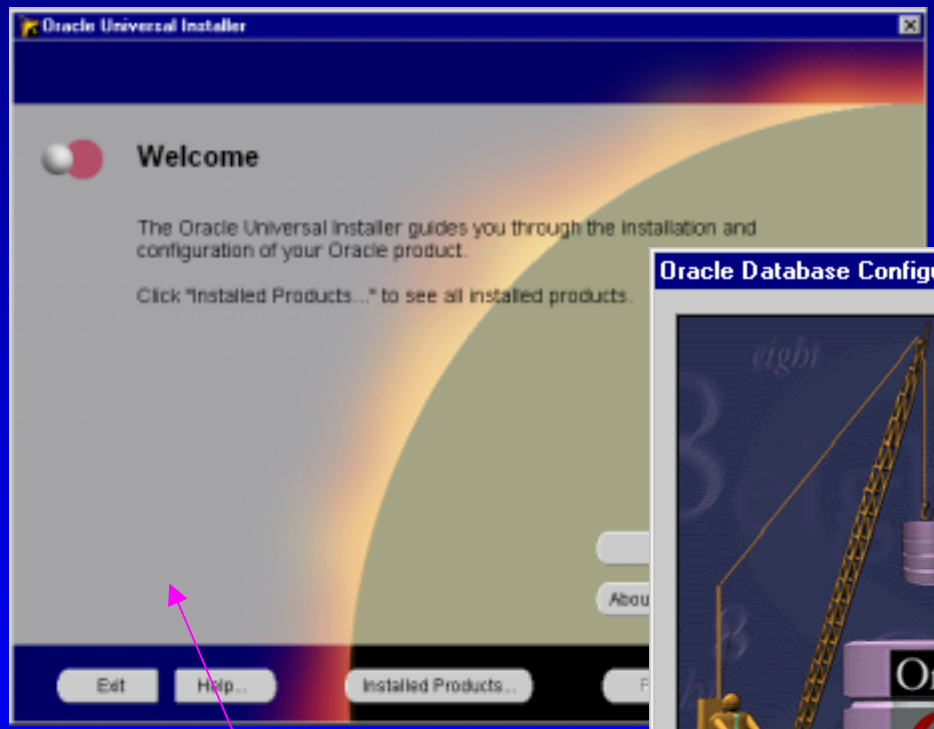
This example is implemented using MATLAB Web Server and Database Toolbox. It uses the web as a user interface, retrieves data from an Oracle database, performs an FFT on the data, and visualizes the data.

Outline

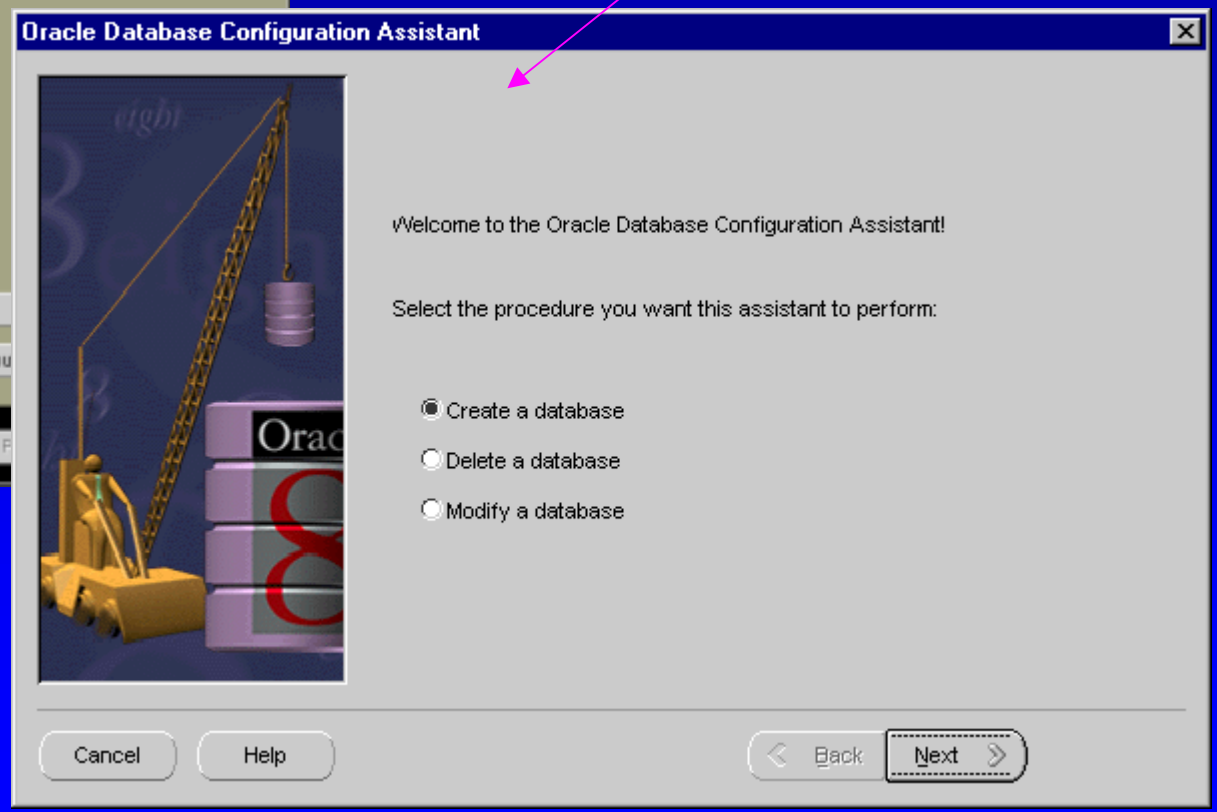
- Step 1: Prepare data using Oracle database
- Step 2: Write the input page in HTML
- Step 3: Write the processing code (M-file)
- Step 4: Write the template output page in HTML.
- Step 5: Configure matweb.conf.
- Step 6: Run your application.

Step 1.1: Prepare data using Oracle database

1.1: Create and name your database *service* (You have to do this in the supervisor mode.) In this example, the database name is IPSDATA.



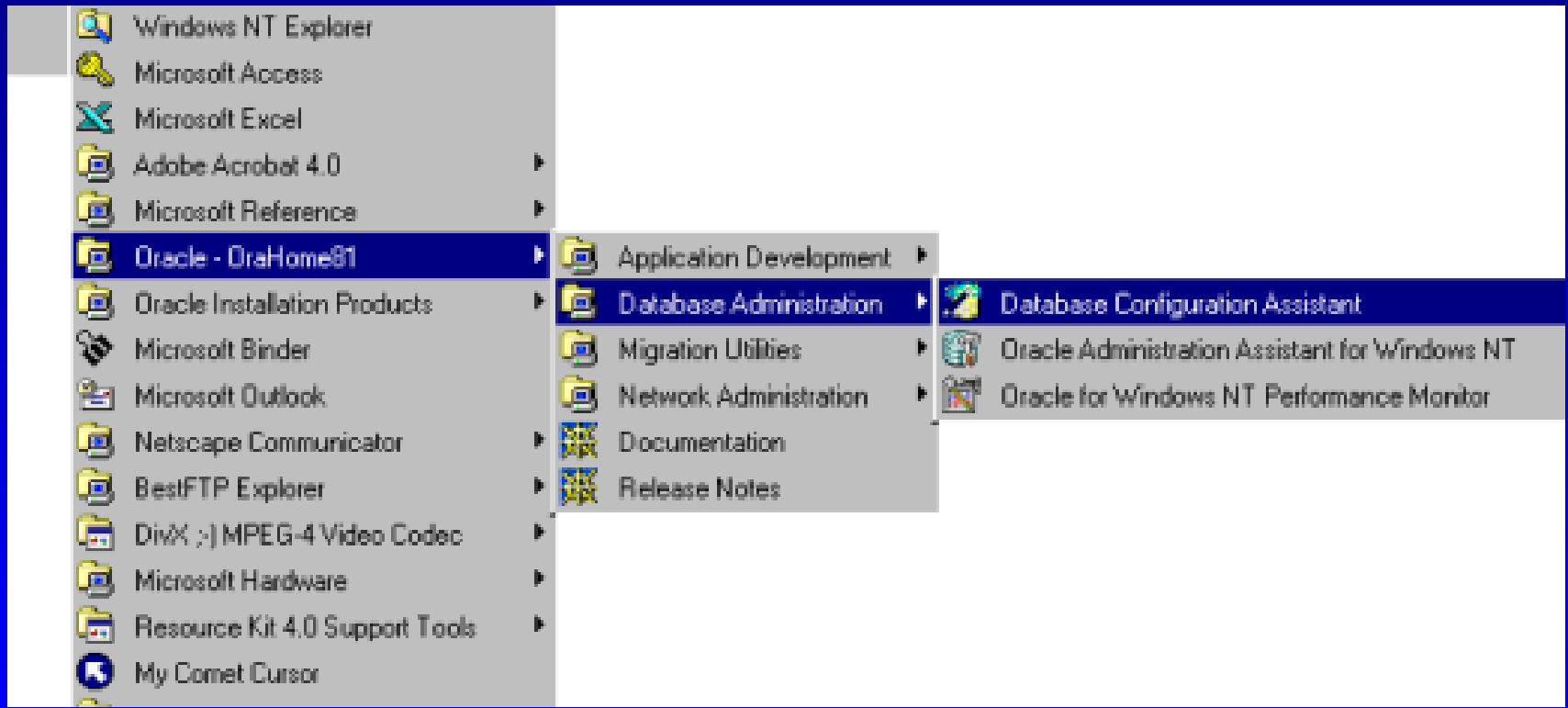
(2) …or you can do this using Oracle Database configuration Assistant after you install Oracle.



(1) Either do this during the installation process. (The Oracle installer will ask you this question.), or…

Step 1.1(cont): Prepare data using Oracle database

1.1: Create and name your database service (You have to do this in the supervisor mode.)

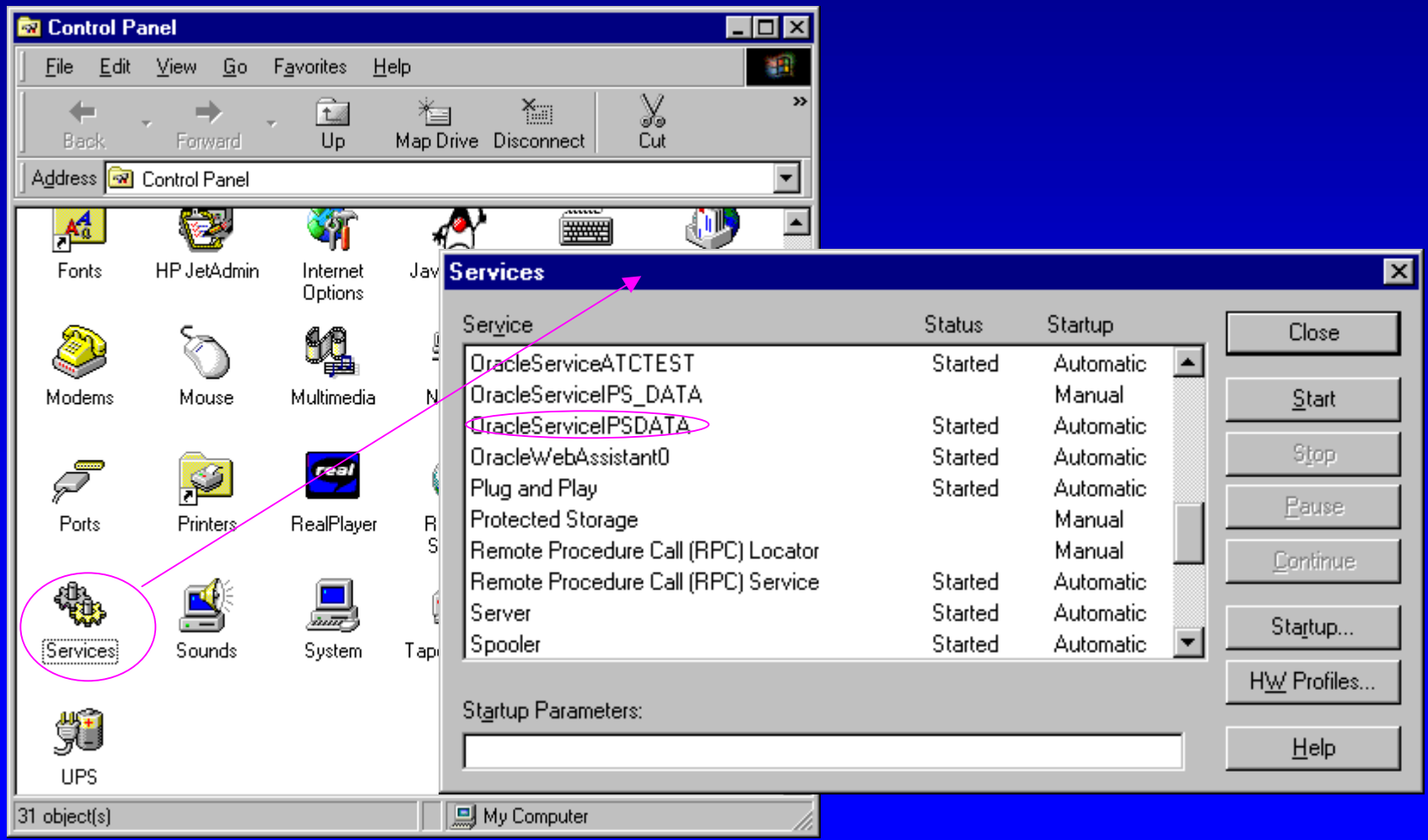


This is how you run the Oracle Database configuration Assistant.

Note: Typically, pre-existing applications don't need to create a database service. Instead just access existing database(s) using a username and password.

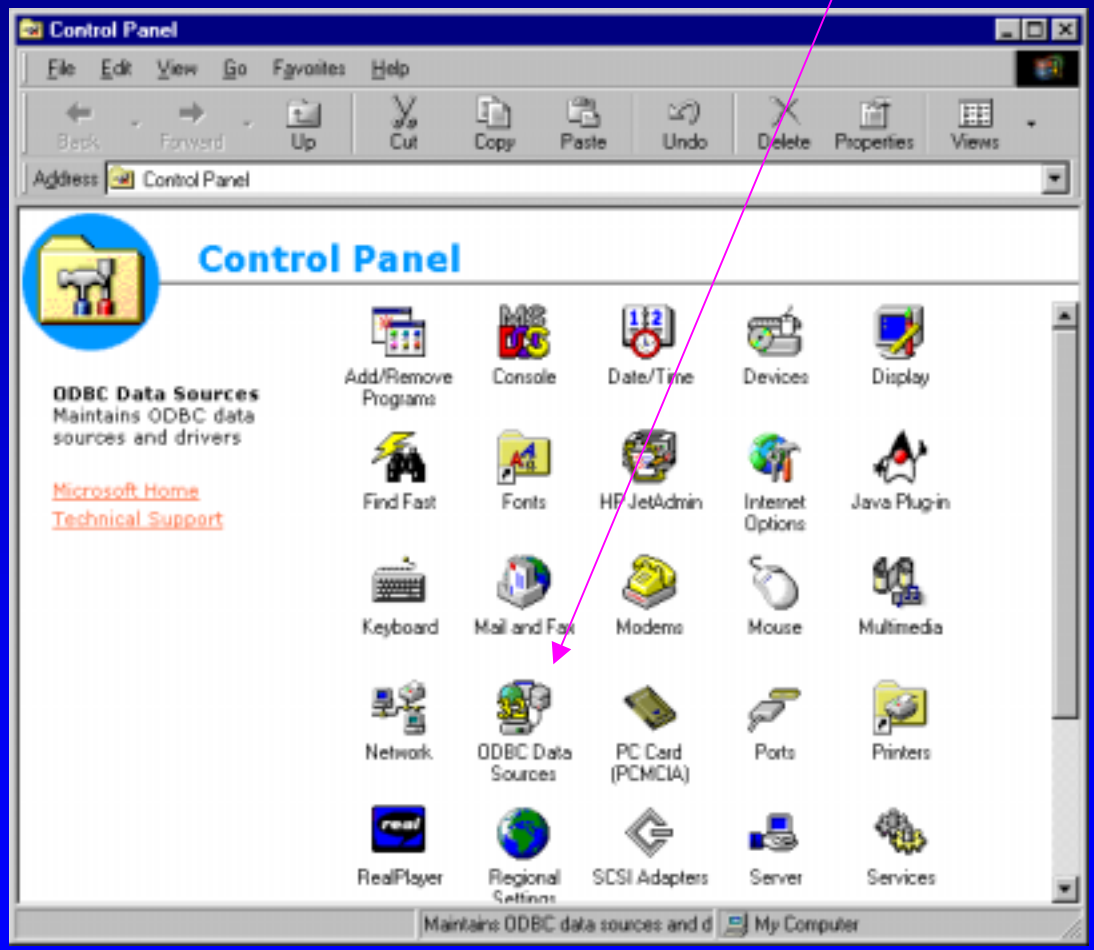
Step 1.2: Preparing the data using an Oracle database

1.2: Invoke the control panel and start this database service.



Step 1.3: Preparing the data using an Oracle database

1.3: Invoke the control panel and double click ODBC Data Sources.

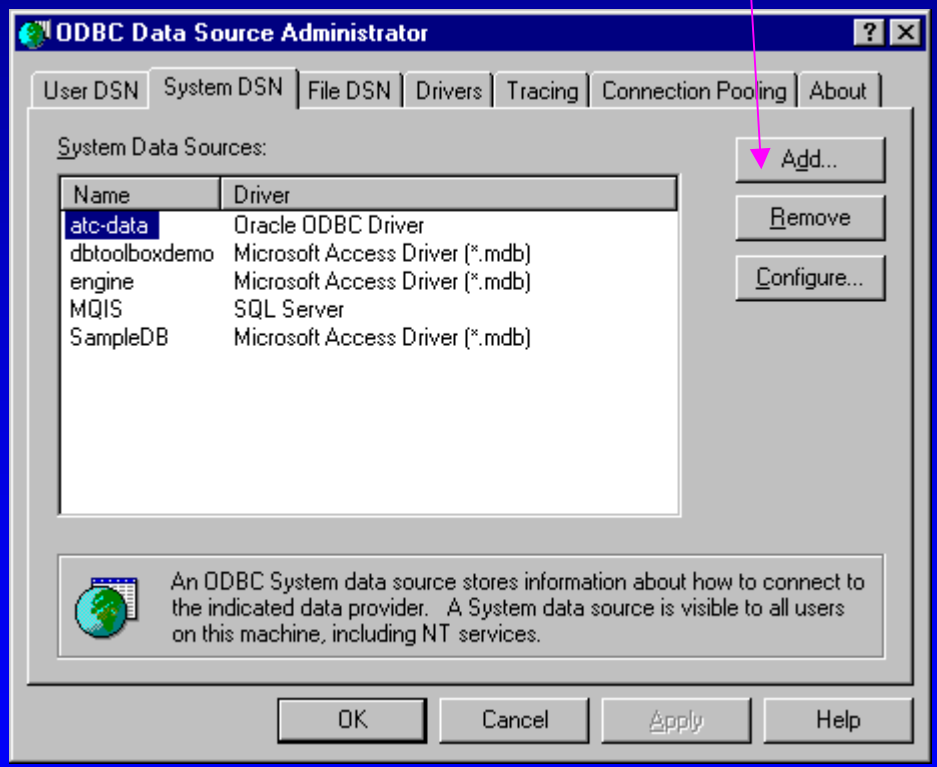


This example is implemented on a WinNT platform.

1.3 --- 1.6 Setup the data source

Step 1.4: Preparing data residing in an Oracle database

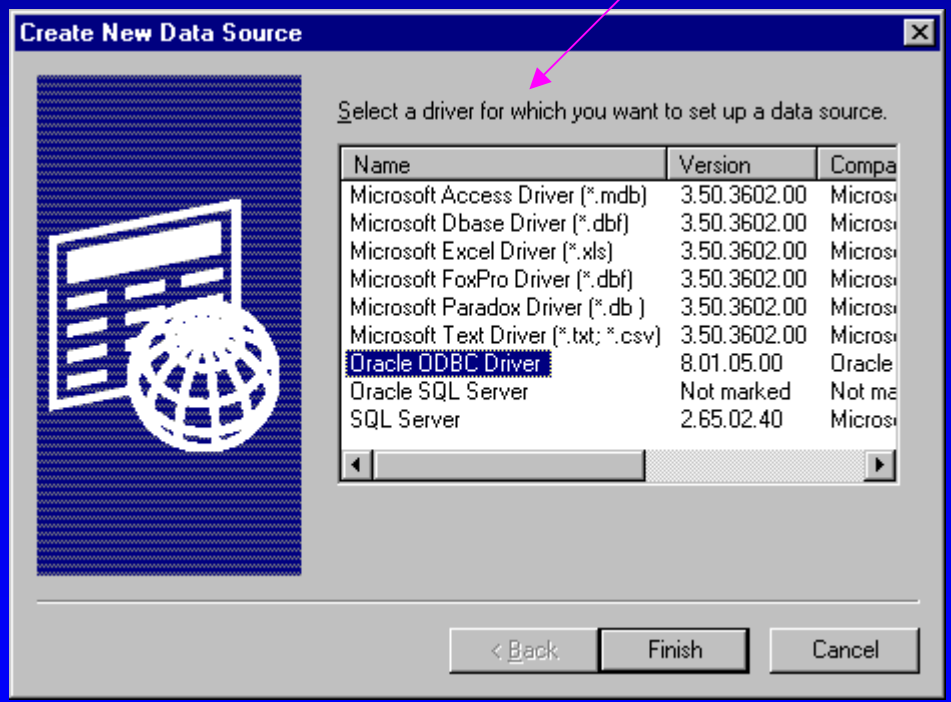
1.4: Select the System DSN tab and click Add.



In order to let the MATLAB Web Server access the data source, you have to specify the data source as a System DSN.

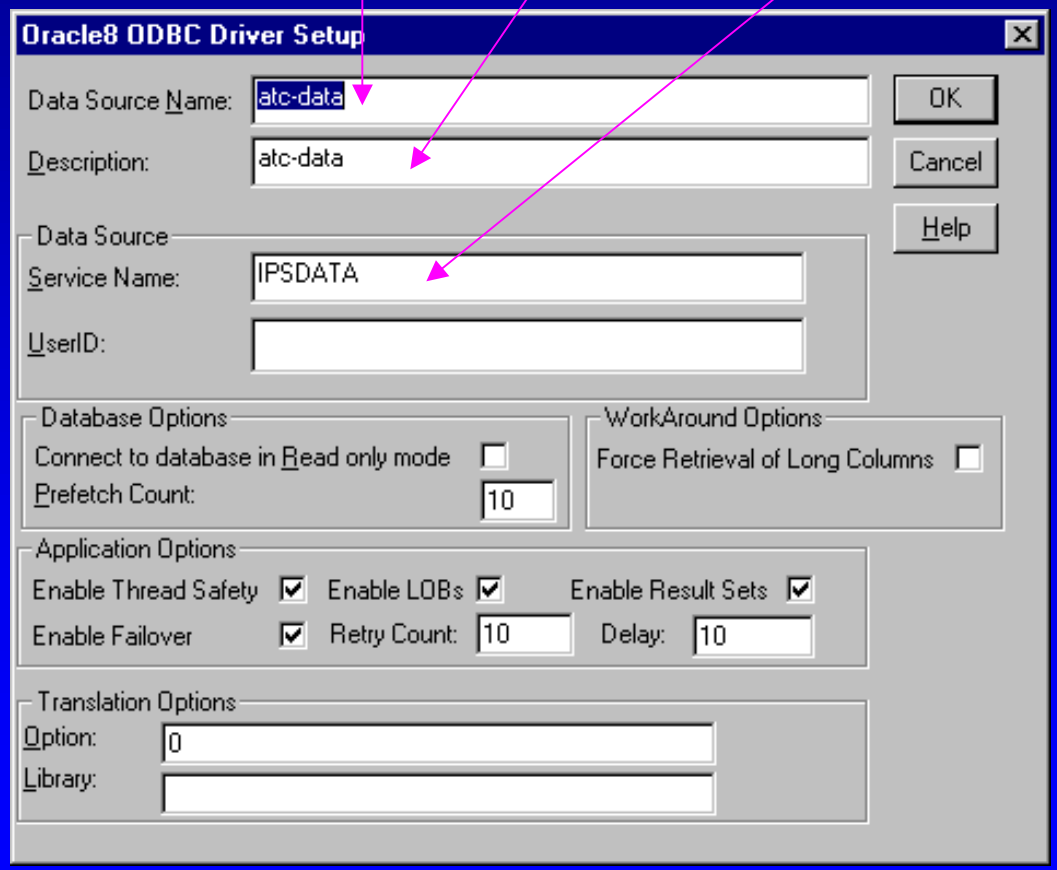
Step 1.5: Preparing data residing in an Oracle database

1.5: Select the ODBC driver. In this case, choose Oracle ODBC Driver.



Step 1.6: Preparing data residing in an Oracle database

1.6: Provide a Data Source Name, Description, and Service Name.

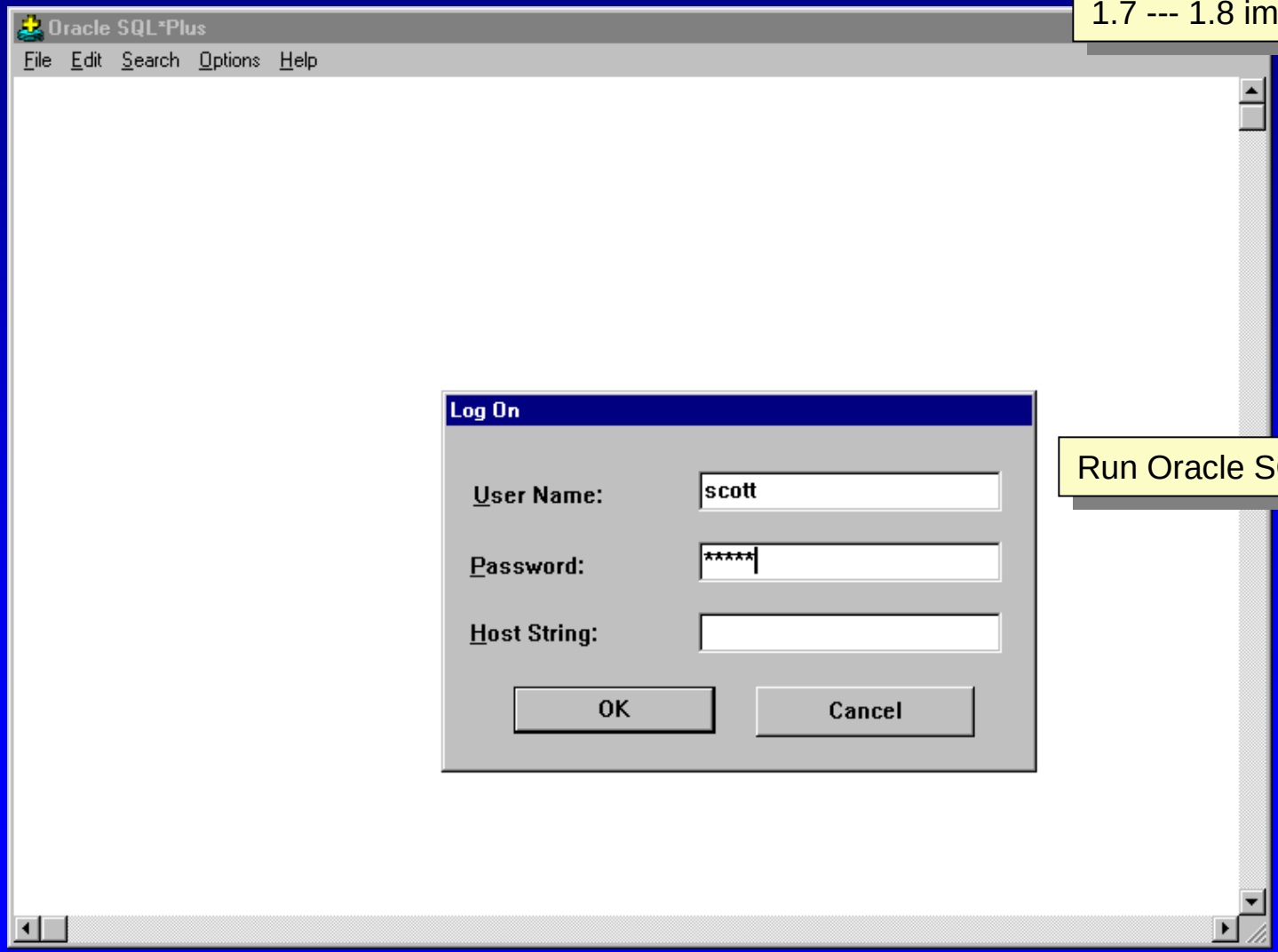


Note: this differs from use of Access in that, instead of specifying a database file, you specify a service name.

Step 1.7: Preparing data residing in an Oracle database

1.7: Import the ATC data into Oracle.

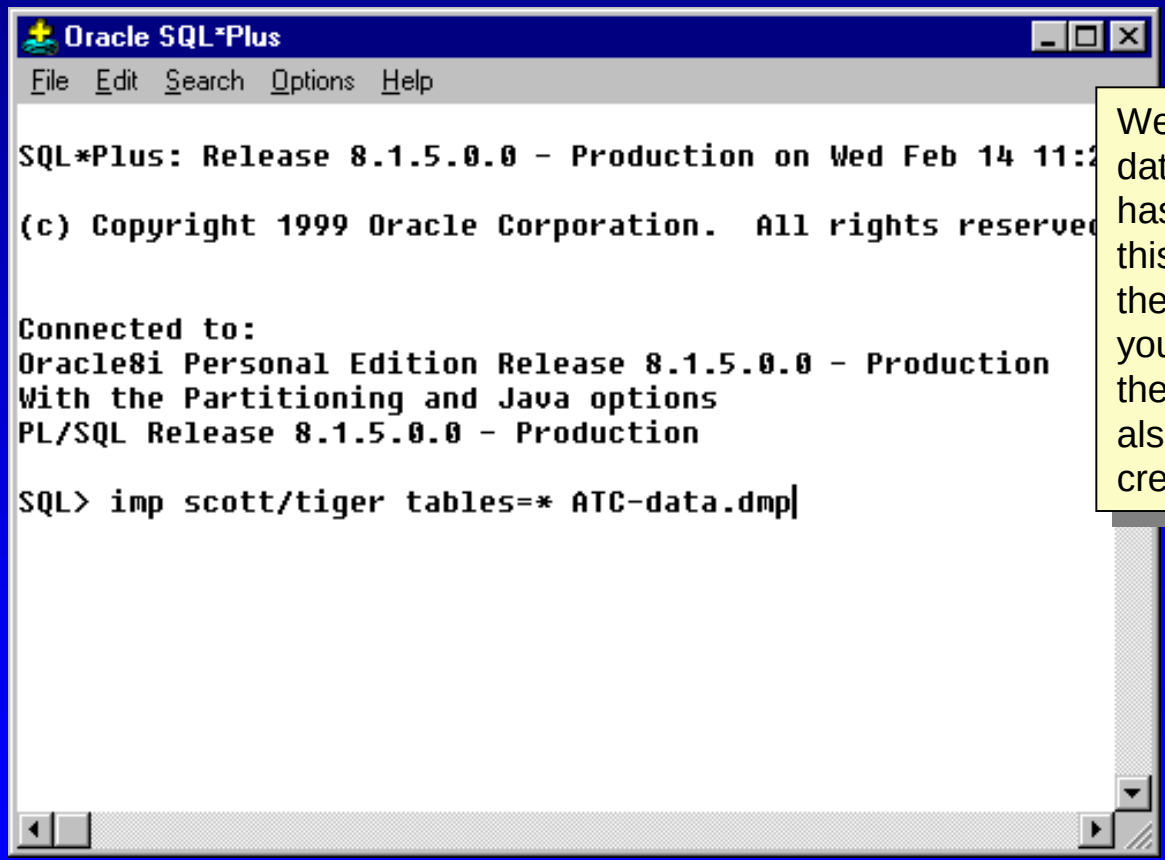
1.7 --- 1.8 import the data



Run Oracle SQL *Plus.

Step 1.8: Preparing data residing in an Oracle database

1.8: Import the ATC data into Oracle.

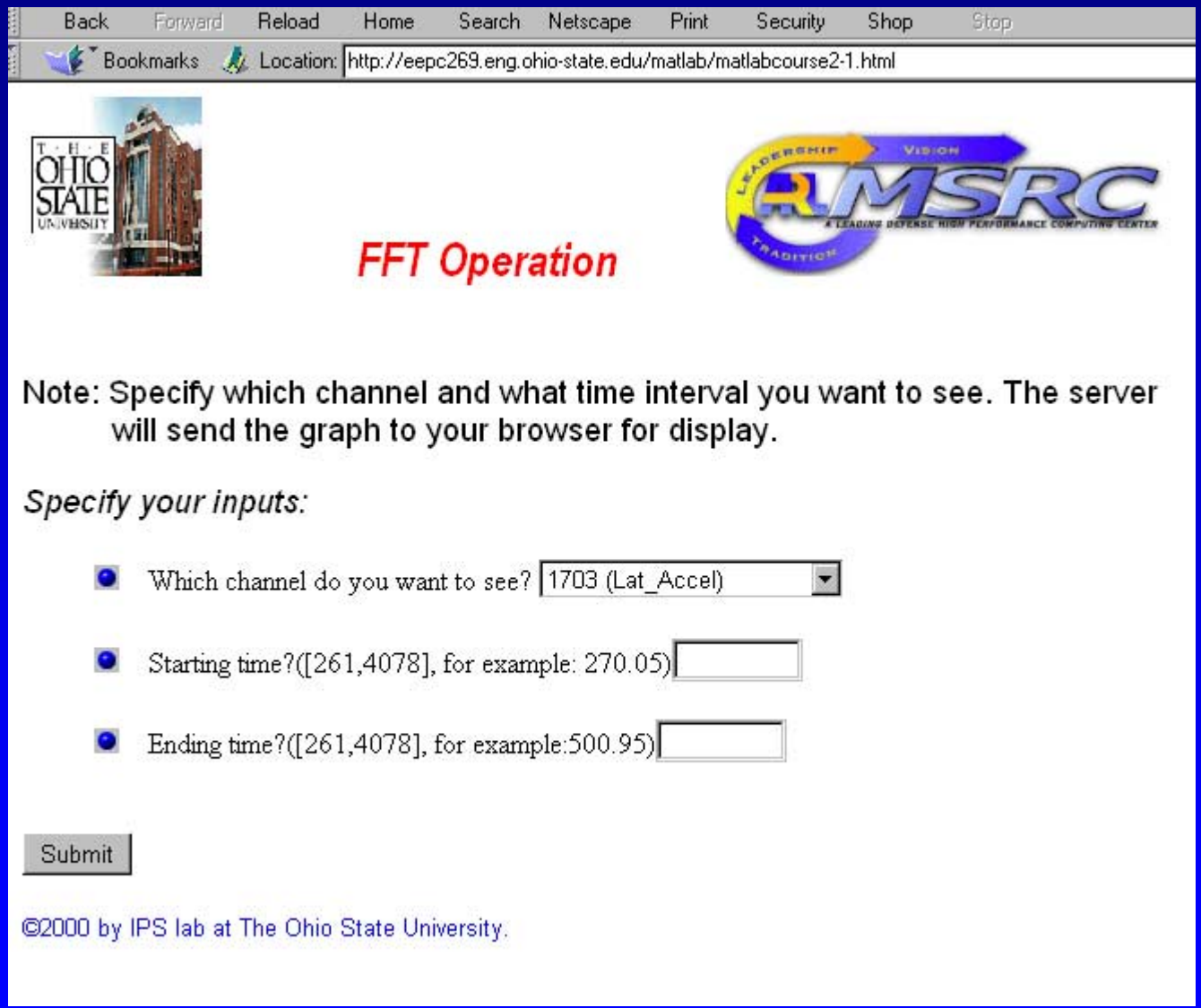


We obtained file 'ATC-data.dmp' from ARL. Oracle has an import utility to import this file into Oracle. If you have the data in a text file format, you can use SQL*Loader to do the job. Of course, you can also use SQL commands to create tables.

Step 1 Summary: Prepare data using Oracle database

If you want to learn more about Oracle and SQL, please check: *Oracle/SQL Tutorial* by Michael Gertz. You can download this tutorial at http://www.db.cs.ucdavis.edu/teaching/sqltutorial/

Step 2: Design the input page



Step 2: Code the input page

```
<form action="/cgi-bin/matweb.exe" method="POST">

<input type="hidden" name="mlmfile" value="matlab2">

<blockquote><img SRC="ball.gif" height=14 width=14>    Which
channel do you want to see? <select name=channel_id><option>1702 (Control_
(Lat_Accel)<option>1704 (Brake_ON)<option>1705 (Inter_Accel_V)<option>1706
(Rear_Accel_V)<option>1707 (Engine_Speed)<option>1708 (Engine_on_hrs)<option>17
(Rd_spd_low_ON)<option>1710 (Brake_On_Total)<option>1711 (Road_Speed)<option>1
(Odometer)<option>1713 (Veh_Motion_hrs)<option>1714 (Low_Spd_Total)<option>1715
(Over_Spd_Total)<option>1716 (Lat_over_Total)<option>1717 (Lat_Accel_&lt;_Tot)<
(Over_Speed_ON)<option>1719 (Accels_ON_light)<option>1720 (Cntrl_Arm_Total)<opt
(Inter_Acel_Total)<option>1722 (Rear_acel_Total)</select></blockquote>

<blockquote><img SRC="ball.gif" height=14 width=14>    Starting
time?([261,4078], for example: 270.05)
<input type="text" size="8" maxlength="8" name="start_time"></blockquote>

<blockquote><img SRC="ball.gif" height=14 width=14>    Ending
time?([261,4078], for example:500.95)

<input type="text" size="8" maxlength="8" name="end_time"></blockquote>
<p><br><input type="submit" name="Submit" value="Submit">
<br></form>
```

call MATLAB web server

M-file name

Select box

Starting time input

Ending time input

Submit button

Step 3: Code the M-file

Part 1 of 4

```
function retstr = matlab2(instruct,outfile)

% parse the input parameters from the input page

channel_id = instruct.channel_id(1:4);
start_time = instruct.start_time;
end_time = instruct.end_time;

% Initialize the return string.

retstr = char('');

sql_cmd = sprintf('select time_interval,channel_value from test_data where channel_id =
  channel_id,instruct.start_time,instruct.end_time);  % construct the query command

feature('dispatchjava',1);    % something you have to run before using Database toolbox
conn = database('atc-data','scott','tiger'); % construct the database connection
curs = exec(conn,sql_cmd); %run the query

curs = fetch(curs);
data = curs.data;  % get the data

x=str2double(data(:,1)); % read the time points
y=str2double(data(:,2)); % read the channle value
```

Extract the input parameters (Uses the names from the input page)

Uses sprintf to generate query methods

Construct the database connection and do the query

Read data

Step 3: Code the M-file

Part 2 of 4

```
x=str2double(data(:,1)); % read the time points
y=str2double(data(:,2)); % read the channle va

% draw the original signal

cd(instruct.mldir);
wscleanup('ml*result1.jpg',1);
f=figure('visible','off');
plot(x,y);
xlabel('Time');
ylabel('Channel value');
tt = sprintf('Channel %s',channel_id);  % construct the title for the plot
title(tt);
grid;

% Save the plot to a JPEG image file

mlid = getfield(instruct, 'mlid');
fn = sprintf('%sresult1.jpg', mlid);
wsprintjpeg(f,fn);

% Assign the JPEG filename to one output parameter.

outstruct.GraphFileName1 = sprintf('/matlab/%sresult1.jpg',mlid);
```

Read the data

Delete files 'ml*result2.jpg' which are older than 1 hour.

Generate a figure handle but without really showing the figure.

Plot the original signal.

Save the plot to a file and assign the filename to the output parameter.

Step 3: Code the M-file

Part 3 of 4

```
outstruct.GraphFileName1 = sprintf('/matlab/%sresult1.jpg',mlid);

% close the connection

close(curs);
close(conn);

% draw the processed signal

wscleanup('ml*result2.jpg',1);
f=figure('visible','off');

% process and plot the FFT result o

yfft = fftshift(log(abs(fft(y,256))
xx = 1:256;
xx = (xx - 129)/128;
plot(xx,yfft);
xlabel('Frequency (* PI)');
ylabel('Magnitude(Logorithm)');
tt = sprintf('Channel %s',channel_id);
title(tt);
grid;

fn = sprintf('%sresult2.jpg', mlid);
wsprintjpeg(f,fn);
outstruct.GraphFileName2 = sprintf('/matlab/%sresult2.jpg',mlid);
```

Close the database connection

Delete files 'ml*result2.jpg' which are older than 1 hour.

Generate a figure handle but without really showing the figure.

Compute FFT and plot the result.

Save the plot to a file and assign the filename to the output parameter.

Step 3: Code the M-file

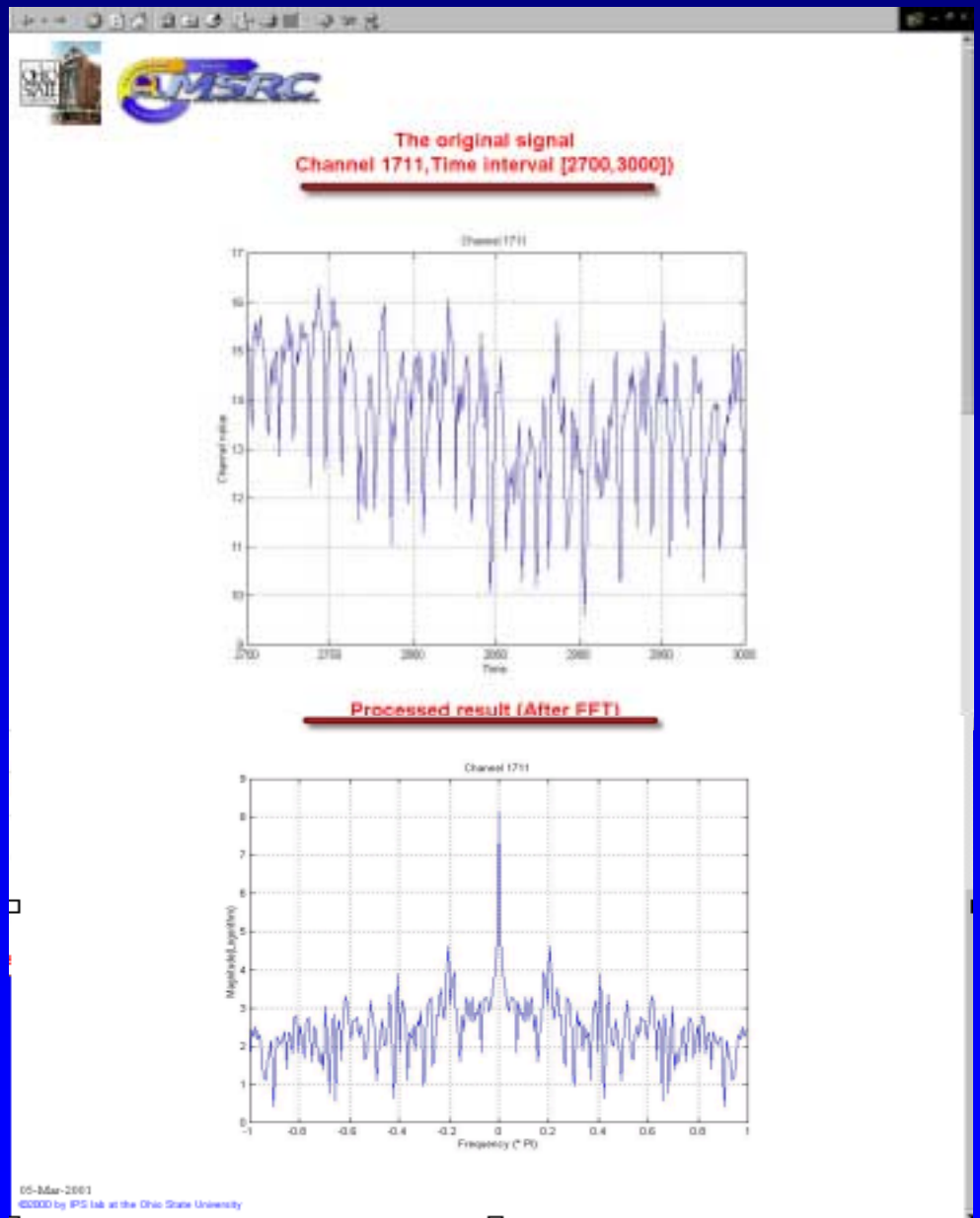
Part 4of 4

```
wsprintjpeg(1,fn);
outstruct.GraphFileName2 = sprintf('/matlab/%sresult2.jpg',ml

% Assign the other output parameters

outstruct.date = date;
outstruct.channel_id = channel_id;
outstruct.start_time = start_time;
outstruct.end_time = end_time;

% Generate the output page from the template output page

templatefile = which('matlabcourse2-2.html');
if (nargin == 1)
   retstr = htmlrep(outstruct, templatefile);
elseif (nargin == 2)
   retstr = htmlrep(outstruct, templatefile, outfile);
end
```

Assign the other output parameters, which will be used in the output page.

Generate the output page based on the template output page 'matlabcourse2-2.html'

Step 4: Code the template output page

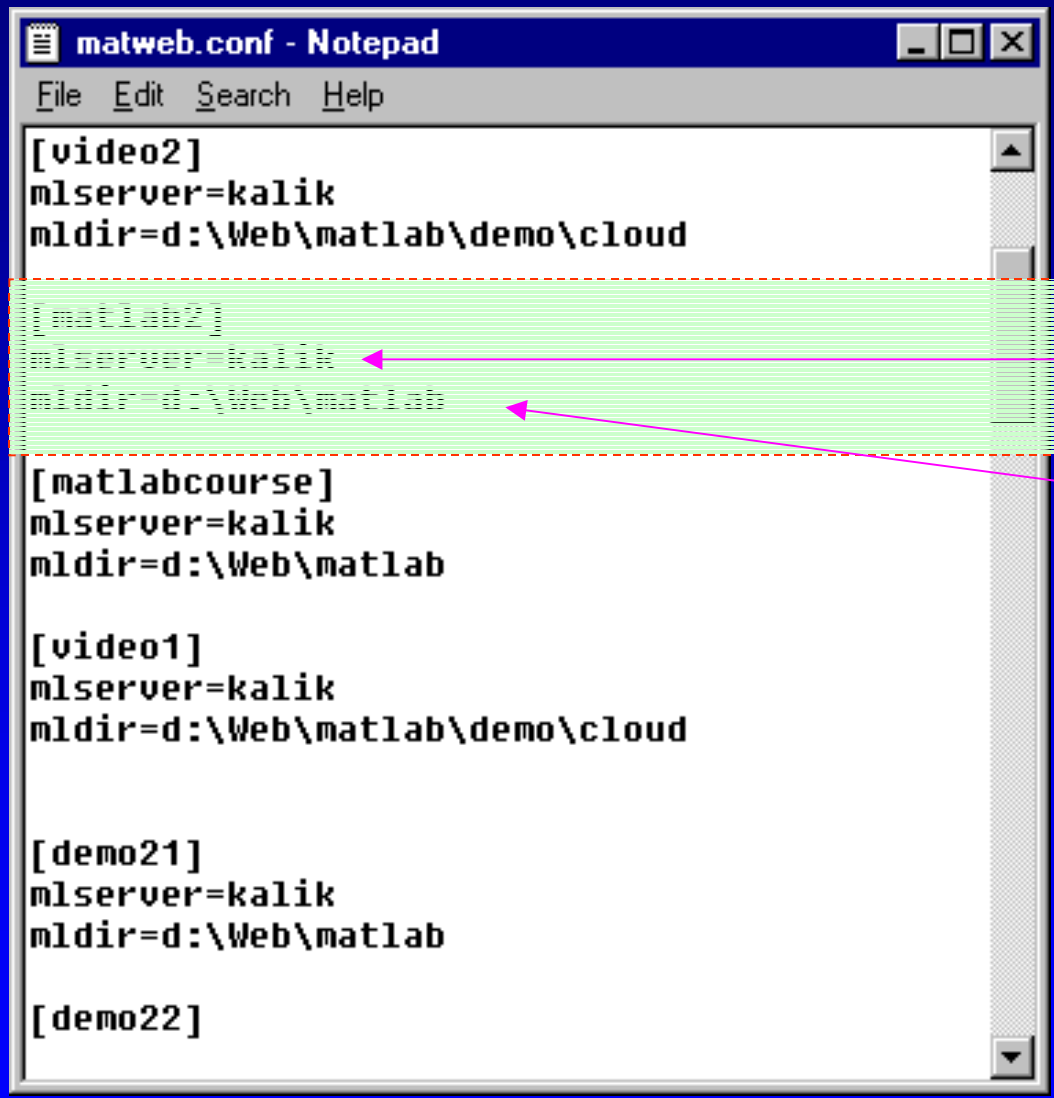


Step 4: Write the template output page

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.74 [en] (WinNT; U) [Netscape]">
   <title>FFT Operation</title>
</head>
<body bgcolor="#FFFFFF">
<img SRC="/matlab/logo-rt.gif" height=100 width=100>    
<img SRC="/matlab/arl.gif" height=83 width=245>
<center><b><font face="Arial"><font color="#FF0000"><font size=+2>The original
signal</font></font></font></b>
<br><b><font face="Arial"><font color="#FF0000"><font size=+2>Channel $channel_id$,Time
interval [$start_time$,$end_time$])</font></font></font></b>
<br><img SRC="/matlab/bar.jpg" height=30 width=450>
<p><img SRC="$GraphFileName1$" BORDER=0 >
<br><b><font face="Arial"><font color="#FF0000"><font size=+2>Processed
result (After FFT)</font></font></font></b>
<br><img SRC="/matlab/bar.jpg" height=30 width=450>
<p><img SRC="$GraphFileName2$" BORDER=0 ></center>

<p>$date$
<br><font face="Arial"><font color="#0000FF"><font size=-1>&copy;2000 by
IPS lab at the Ohio State University</font></font></font>
</body>
</html>
```

The variables delimited by $ will be replaced by the processed result

Step 5: Modify matweb.conf



Machine name for MATLAB Web Server

Working directory

Note: file matweb.conf is located in the directory aliased by /cgi-bin or equivalnent.

Step 6: Run your program

In this case,
http://eepc269.eng.ohio-state.edu/matlab/matlabcourse2-1.html

More examples

Check our website

Http://eepc269.eng.ohio-state.edu/matlab/